collaborative Protection Profile for Dedicated Security Component

Version 2.1, November 17, 2025

Acknowledgements

This collaborative Protection Profile (cPP) was developed by the Dedicated Security Components international Technical Community (DSC-iTC) with representatives from Industry, Information Technology Security Evaluation Facilities (ITSEFs), and International Common Criteria schemes. The organizations that directly contributed to the development of this cPP include:

Industry

Google, LLC.

Samsung Electronics Co., Ltd.

Common Criteria Test Laboratories

atsec Information Security

Det Norske Veritas and Germanischer Lloyd (DNV GL)

International Common Criteria Schemes

National Information Assurance Partnership (NIAP)

UK IT Security Evaluation and Certificate Scheme (NCSC)

Chapter 1. Preface

1.1. Objectives of Document

This document presents the Common Criteria (CC) collaborative Protection Profile (cPP) to express the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) for a Dedicated Security Component (DSC). The Evaluation Activities that specify the actions an evaluator performs to determine if a product satisfies the SFRs and SARs captured within this cPP are described in the Evaluation Activities for Dedicated Security Component cPP Supporting Document [DSC SD].

The DSC international Technical Community (iTC) designed the DSC cPP as a standalone PP so that vendors may evaluate a DSC once and re-use this evidence across multiple devices that contain identical DSCs. Vendors may also combine this cPP with a platform solution cPP for CC consumers.

1.2. Scope of Document

The scope of the cPP within the development and evaluation process is described in the Common Criteria for Information Technology Security Evaluation [CC]. In particular, a cPP defines the IT security requirements of a generic type of Target of Evaluation (TOE) and specifies the functional and assurance security measures that the ITSEF must apply to the TOE to demonstrate that it meets the cPP's stated requirements [CC1, Annex B].

1.3. Intended Readership

The target audiences of this cPP are developers, CC consumers, system integrators, evaluators, and schemes.

1.4. Related Documents

Table 1. Related Documents

ID	Title	Source
[CC1]	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, CCMB-2022-11-001, CC:2022 Revision 1, November 2022	https://www.commoncriteriaportal.org/cc/index.cfm
[CC2]	Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, CCMB-2022-11-002, CC:2022 Revision 1, November 2022	https://www.commoncriteriaportal.org/cc/index.cfm
[CC3]	Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components, CCMB-2022-11-003, CC:2022 Revision 1, November 2022	https://www.commoncriteriaportal.org/cc/index.cfm

ID	Title	Source
[CC4]	Common Criteria for Information Technology Security Evaluation, Part 4: Framework for the specification of evaluation methods and activities, CCMB-2022-11-004, CC:2022 Revision 1, November 2022	https://www.commoncriteriaportal.org/cc/index.cfm
[CC5]	Common Criteria for Information Technology Security Evaluation, Part 5: Pre-defined packages of security requirements, CCMB-2022-11-005, CC:2022 Revision 1, November 2022	https://www.commoncriteriaportal.org/cc/index.cfm
[CEM]	Common Methodology for Information Technology Security Evaluation, CCMB-2022-11-006, CC:2022 Revision 1, November 2022	https://www.commoncriteriaportal.org/cc/index.cfm
[CC-E&I]	Errata and Interpretation for CC:2022 (Release 1) and CEM:2022 (Release 1), 002, Version 1.1, July 22, 2024	https://www.commoncriteriaportal.org/cc/index.cfm
[DSC SD]	Supporting Document: Evaluation Activities for collaborative Protection Profile for Dedicated Security Component: Mandatory Technical Document, Version 2.1, November 17, 2025	https://dsc-itc.github.io

1.5. Revision History

Version	Date	Description
1.0	9/10/20	Initial publication
2.0	October 28, 2024	Final publication for v2.0 (conversion to asciidoc, respond to CCDB comments)
2.1	November 17, 2025	Final publication for v2.1 (post-quantum cryptography, respond to NIAP comments)

Table of Contents

List of Figures

List of Tables

Chapter 2. PP Introduction

2.1. PP Reference Identification

PP Reference: collaborative Protection Profile for Dedicated Security Component

PP Version: 2.1

PP Date: November 17, 2025

2.2. TOE Overview

The Target of Evaluation (TOE) is a Dedicated Security Component (DSC). In the context of this cPP, a DSC is the combination of one or more hardware component(s) and its controlling OS or firmware. The firmware should be dedicated to providing the encompassing platform with services for the provisioning, protection, and use of Security Data Objects (SDOs), which are composed of Security Data Elements (SDEs) such as keys, identities, attributes. See Figure 1 for an example of a TOE representation.

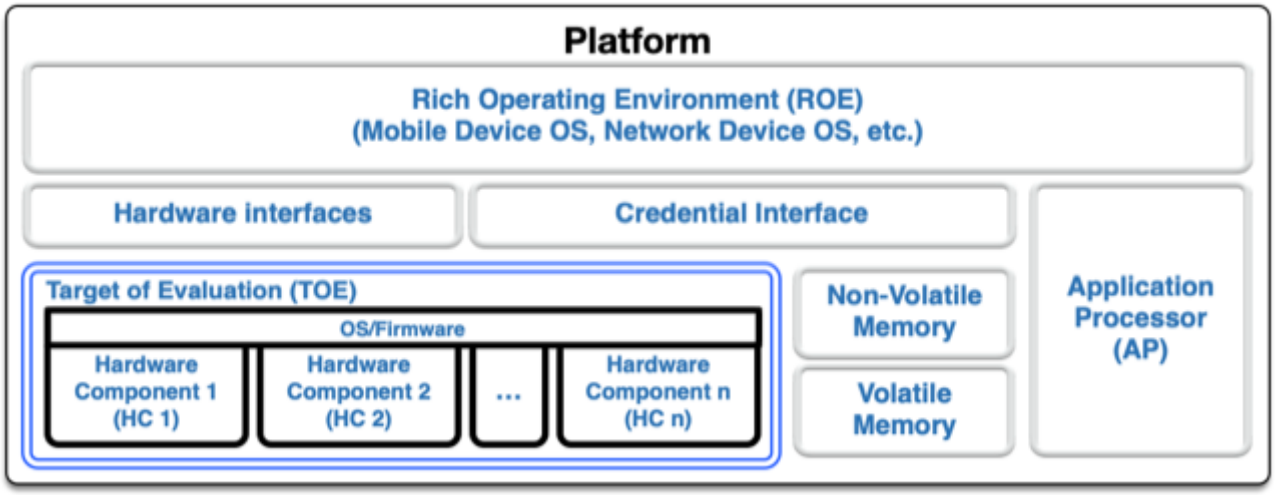


Figure 1. Representation of the Target of Evaluation (TOE)

The TOE should be one or more discrete and embedded hardware components that provide well-scoped security functions that are physically inaccessible directly from the rich operating system. The DSC TOE would consist of isolated firmware and circuitry capable of executing well-defined commands against SDEs/SDOs within the TOE and outside the TOE across restricted interfaces. The DSC TOE is not intended to be a discrete, separate stand-alone component, but one which is directly embedded into a larger system.

There are many possible configurations for a DSC. Some examples are:

- A DSC may be comprised of a single embedded component within a device, such as a Secure Enclave Processor (SEP) or System on Chip (SoC)
- A multi-component system comprised of a software layer and several hardware components (which may be discrete or embedded), such as a Trusted Execution Environment (TEE)

Other configurations are possible, with the key point being the DSC is embedded within a larger system and is not a discrete component.

These dedicated hardware/software components are isolated components of a larger physical package. Figure 2 below shows a block diagram of a typical example of a DSC TOE with all of its internal components.

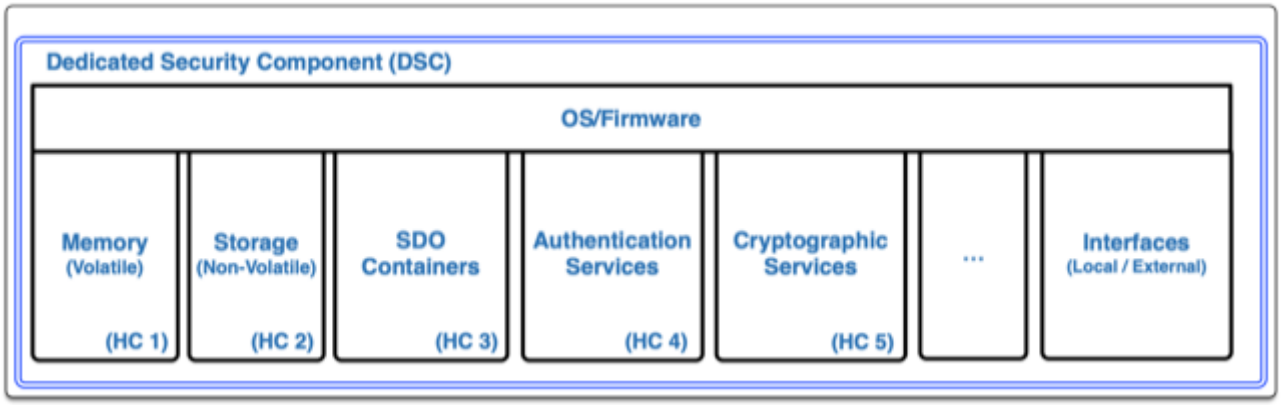


Figure 2. Example of TOE Internal Components

2.2.1. Security Data Objects

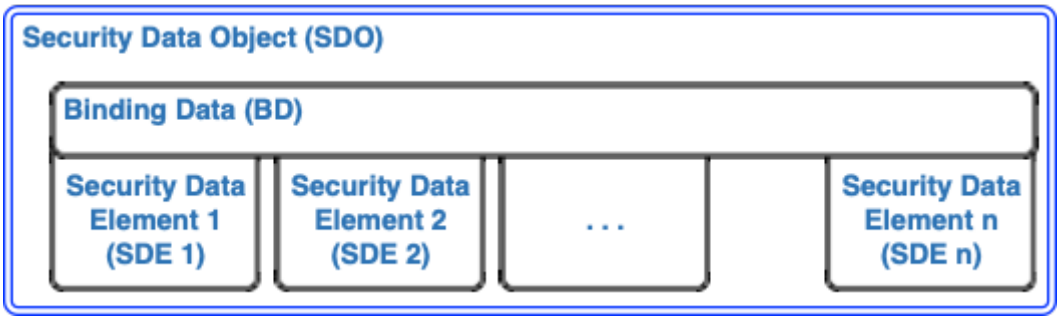


Figure 3. Composition of an SDO

An SDO is created by combining SDEs with some attributes. Each SDE used to create the SDO reaches the DSC in one of the following ways:

- By parsing SDEs received via secure channels (see O.PARSE_PROTECTION).
- By generating the SDEs locally on the DSC as part of the Provisioning service.

An SDO may include one or more SDEs from one or both of these sources. In the Provisioning step, the relevant SDEs are then bound together with a set of attributes resulting in an SDO. Explicit binding occurs when the DSC includes one or more SDEs along with their attributes in a formatted structure to form the SDO. An X.509 certificate is just one example of an SDO (where the signature in the certificate provides the binding of the attributes contained). A DSC protects the integrity of an SDO (see O.DATA_PROTECTION).

Explicit binding may also occur when the DSC wraps an SDO prior to storing it externally. Figure 3 shows an example SDO with binding data used to secure an arbitrary number of SDEs.

Implicit binding may occur by virtue of the location of SDEs within the DSC. An implicit binding may occur for pre-installed SDEs, in which case the DSC restricts the functionality it allows with the SDEs as part of the firmware itself. It may also occur when the contents of certain protected storage

locations carry with them implicit attributes simply by existing in these locations.

Vendors may pre-install keys and other material in the DSC during the manufacturing process, or the DSC may automatically generate keys or other material upon first boot. Since the user (an administrator or client application acting on behalf of a human user) provides no input to these items, the cPP calls these pre-installed SDEs. Pre-installed SDEs have two distinguishing characteristics:

- These keys may persist over a factory reset; and
- They may not be accessible to administrators.

If the SDEs have been erased (e.g. due to a tamper response), then a factory reset may not be possible. Following an initial boot (e.g. first boot by end-user, or following a factory reset), a DSC may generate SDEs unique to an instance of a DSC that are persisted across user sessions. These are considered to be pre-installed SDEs.

Pre-installed SDOs (i.e., SDEs with implicit binding installed by the vendor at manufacturing time) are typically not accessible by non-administrative users of the platform (i.e., client applications) and are reserved for use by the DSC itself to manage its sub-components, keys, and, indirectly, user content. Pre-installed SDOs typically have implicitly bound attributes. Since pre-installed SDOs rarely, if ever, leave the DSC, they may have no formal structure containing attributes. That does not mean these attributes do not exist; only that there exists no structure in which one would find them all in one place.

The DSC may allow the modification of attributes for pre-installed SDOs. One example would be the authorization value necessary to use the SDO. Obviously, the vendor may have a strong desire to keep the users of the DSC from changing the SDE itself, or deleting it. They could allow administrators to hide the SDO, but not delete it for the sake of factory resets.

Another case of implicit binding occurs when a DSC reserves a bank of user-accessible registers with common attributes. The bank contains one or more registers, usually all of the same size. Again, the functionality within the firmware determines the attributes especially when the function applies only to one or more members of the bank of reserved registers. Without the benefit of a structure with explicit attributes, the DSC relies on the firmware to enforce the policies inherent to the attributes associated with a bank of registers; for example, the DSC firmware implicitly binds the common attributes to the bank of registers.

An SDO held in the DSC may be exported (propagated) only if it is either in a wrapped form (i.e. with confidentiality and integrity of the SDO protected by a cryptographic key-based operation), or if it is transmitted over a secure channel (protecting confidentiality, integrity and optionally authenticity of the receiving endpoint).

2.2.2. Services

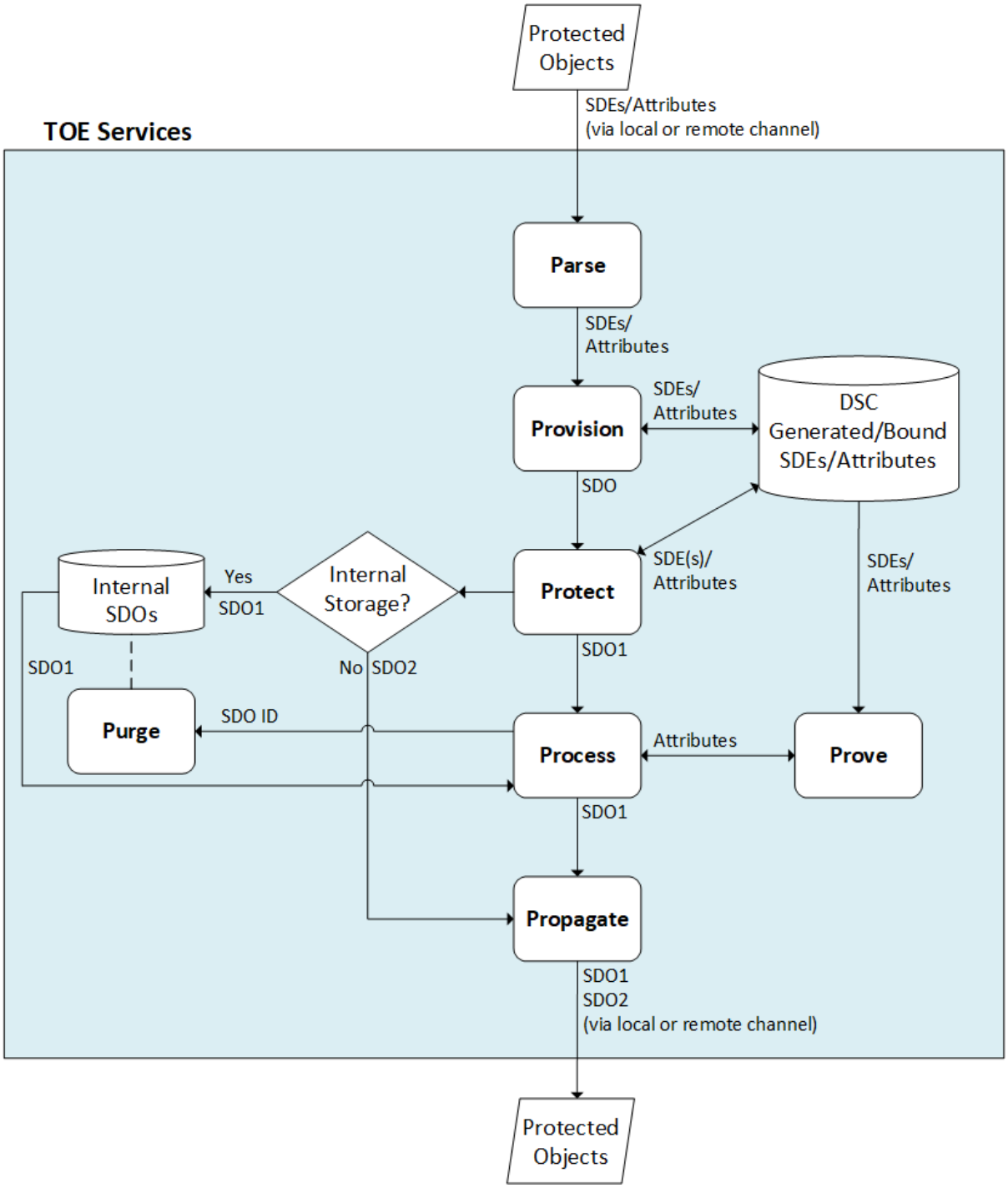


Figure 4. Services Provided by the TOE

The labels in Figure 4 refer to the following:

- SDE: Security Data Element
- SDO: Security Data Object (composed of SDEs and attributes)
- SDO ID: Unique identifier for an SDO
- SDO1: SDO that is modified or is a reference to original SDO
- SDO2: SDO that is bound to the DSC but stored outside of it

DSCs provide seven core security services to a platform as illustrated in Table 2.

Table 2. Core Security Services

Service	Description
Parse	The DSC shall ingest pre-installed keys, credentials, tokens as SDEs and where applicable, with attributes to form SDOs, from trusted components or services external to its boundary either across a secured channel or in a manner that the objects are protected for use only by the DSC.
Provision	The DSC shall create SDOs from parsed or generated SDEs and attributes using binding mechanisms to apply integrity protection to the SDEs together with their attributes.
Protect	The DSC shall manage protected storage for all SDOs. DSCs may implement local storage internal to the DSC boundary or utlilize external storage outside the DSC boundary. A DSC shall maintain the integrity and confidentiality (if required) of SDOs stored both inside and outside the boundary.
Process	The DSC shall modify and use SDOs or their attributes on behalf of authorized entities. The Process service shall coordinate with the Protect service for storage of the SDOs while not in use and shall collaborate with the Prove service to authenticate the requesting entity and validate their authorization for access to the SDO in the requested mode. The Process service shall submit an SDO to the Purge service when it is no longer needed by the platform.
Prove	The DSC may attest to a remote entity that the DSC is currently in a specific state. During this process, the DSC shall use the appropriate attributes or authentication tokens (such as nonces, digital signatures, etc.) to enable the remote entity to verify the authenticity of the source of the evidence.
Purge	When the platform no longer needs an SDO, the DSC shall execute a mechanism for destroying the SDO by permanently removing it from the DSC to protect against unauthorized recovery.
Propagate	If an SDO is required by or allowed to be used by a remote peer, the DSC shall ensure that the SDO is exported only as a protected object or is transmitted over a trusted channel.

2.2.3. Roots of Trust

This collaborative Protection Profile (cPP) assumes a DSC will contain a Root of Trust (RoT) that is comprised of the compute engine, one set of firmware code, and pre-installed SDOs, including a unique identity bound to the hardware. The firmware code may be immutable, or it may be mutable but with controlled, authenticated, and authorized updates allowed to ensure continued integrity of the RoT. This code may provide one or more RoT services, such as a RoT for Measurement, Verification, or Reporting. The unique identity bound to the hardware should be immutable and third parties should be able to authenticate the manufacturer of the Root of Trust through its unique identity (e.g., the unique identity may be a credential signed by the manufacturer).

2.2.4. DSC Characteristics

The security functional requirements rely on the following characteristics of the DSC:

- Subjects
- Roles
- Objects
- Security Attributes
- Operations

Subjects: The following list contains the fundamental actors in the expected operational use cases of the DSC. The first three are active actors, while the fourth is usually passive but could be active.

- S.DSC - DSC with security attribute DSC.ID, which is the identity of the DSC
- S.Admin - Admin (an authorized administrator with special privileges) security attribute - See Section 2.2.5 for more discussion on administrator roles.
- S.CApp - Client Application (CApp) (i.e. an authorized user or an application with a verifiable identity) with security attribute CApp.ID - See Section 2.2.5 for more discussion on user roles.
- S.EPS - External Platform Storage (EPS) (e.g. transient SDE/SDO source and destination, in the case of data imported and exported for the sole use inside the DSC). In the case of a passive EPS, the DSC will properly protect the integrity and confidentiality of the objects it stores and retrieves from there. In the case of an active EPS with security attribute EPS.ID, the DSC and EPS may choose to create a secure channel through which they will pass objects back and forth.

Roles: Users of the DSC are assigned to Roles which enumerate the permissions which are granted to the Objects and Operations. See Section 2.2.5 for more discussion on roles.

Objects: The following list contains objects the DSC expects to use during the expected operational use cases.

- OB.P_SDO - Pre-installed SDOs (e.g. DSC.ID) with security attributes listed in the next paragraph.
- OB.T_SDO - Transient SDOs or just SDOs (i.e. SDOs in the DSC currently, but are either ephemeral or are normally stored external to DSC when not in use) with security attributes listed in the next paragraphs. See Section 2.4.1, Section 2.4.3, and Section 2.4.5 for more discussion on keys, which are the primary use cases for SDOs.
- OB.AuthData - Authorization Data (including authentication data, e.g. PINs, passwords, tokens)
- OB.Pstate - Platform State (e.g. measurements and assertions)
- OB.FAACntr - Failed Authorization Attempt Counters
- OB.AntiReplay - Anti-replay tokens (e.g. counters, nonces, etc.)
- OB.Context - Session Context (The DSC may maintain one or more sessions with a CApp involving one or more of SDOs, Authorization Data, Platform State, Failed Authorization Counters, and Anti-Replay Tokens. The DSC may represent internally the state of these objects at any given time in a Session Context) - See Section 2.4.2 for more discussion on sessions.

Security Attributes: The following list contains the minimum security attributes for a DSC. Individual DSCs may implement additional security attributes beyond this (whether they are additional standalone attributes or additional attributes that are associated with SDOs).

- DSC.ID - The DSC identifier. It may also serve as the identifier for the DSC RoT.
- CApp.ID - The Client Application identifier.
- EPS.ID - The External Platform Storage (EPS) identifier. This attribute is optional for a passive EPS (i.e. plain memory that only stores information). If the DSC uses an active EPS to manage storage, then support for this attribute is required.
- SDO.* - The SDO Security Attributes:
  - SDO.ID - SDO Identifier
  - SDO.Type - SDO Type
  - SDO.AuthData - SDO Reference authorization data
  - SDO.Reauth - SDO re-authorization conditions
  - SDO.Conf - SDO confidential SDE list
  - SDO.Export - SDO export flag
  - SDO.Integrity - SDO integrity protection data
  - SDO.Bind - SDO binding data

Operations: The following list contains the expected operations of a DSC.

- OP.Import (See Parse) - The DSC may receive SDOs, SDEs, Authorization Data, Platform State, Anti-Replay Tokens or Session Contexts from the CApp or the EPS. The Admin may also give the DSC Authorization Data.
- OP.Create (See Provision) - The DSC may create SDOs, SDEs, Authorization Data, Platform State, Anti-Replay Tokens, or Session Contexts with authorization from a CApp or Admin.
- OP.Use (See Process) - The DSC may use or perform a cryptographic operation on Pre-Provisioned SDOs, Transient SDOs, SDEs, Authorization Data, Platform State, Anti-Replay Tokens, or Session Contexts with Create authorization from a CApp or Admin. Cryptographic operations may include encryption, decryption, hashing, signature generation, and signature verification.
- OP.Modify (See Process) - The DSC may modify SDOs, SDEs, Authorization Data, Platform State, Anti-Replay Tokens, or Session Contexts with authorization from a CApp or Admin.
- OP.Attest (See Prove) - The DSC may create an attestation of Platform State using an SDO or Pre-Provisioned SDO and Anti-Replay Tokens as authorized by a CApp or Admin respectively.
- OP.Store (See Protect) - The DSC may store SDOs, SDEs, Authorization Data, Platform State, Anti-Replay Tokens, or Session Contexts in protected storage of the DSC. See section 2.4.5 for more discussion on protected storage.
- OP.Export (See Propagate) - The DSC may export SDOs, SDEs, Authorization Data, Platform State, or Anti-Replay Tokens to a CApp or EPS with the proper authorization from the owner of each object. In the case of EPS, the DSC will bind the objects to the DSC in such a way as to deny other DSCs or entities the ability to import, use, modify, attest, store, export, or destroy them. The DSC may export Session Contexts only to an EPS binding it in the same way as above.
- OP.Destroy (See Purge) - The DSC may purge SDOs, SDEs, Authorization Data, Platform State, Anti-Replay Tokens, or Session Contexts in protected storage with proper authorization from the

owner of each object.

2.2.4.1. Concept of Users in DSC

The entities using the DSC will be client applications on the platform. They may be acting as proxies for users or may have identities of their own. The DSC will not be able to distinguish the difference; therefore, the cPP will recognize an entity known as the Client Application (CApp), as the user presenting authentication tokens and authorization values (collectively known as authorization data) to the DSC for the purposes of identity verification and authorization to perform operations.

The term users may be used throughout the cPP as a stand-in for Client Application, but there is not a specific requirement for direct user accounts or users within the system as opposed to the Client Applications.

2.2.5. Roles

As with many systems, rather than managing access rights individually for each "user" of the DSC, access rights are managed through the use of roles. Within the DSC, there are three possible roles that are defined. These roles are defined as:

- ADM-R - Owner Admin role - the administrator role related to the management of the DSC once it has been integrated into a platform.
- MFGADM-R - Manufacturer Admin role - the administrator role related to the management of firmware and key material that form the basis for the root of trust.
- CApp-R - Client Application role - the client role of the DSC that requests and utilizes the functionality provided by the DSC.

Depending on the configuration of the DSC, there may not be a separation of the Admin roles, such that the capabilities of the ADM-R and MFGADM-R roles are combined into a single role. For the purposes of the cPP, unless specifically called out, all administration roles are assumed to be combined and will use ADM-R.

As the DSC is generally a component within a larger system or platform, the roles of the DSC are specific to the DSC. While there may be matching roles between the DSC and its platform, the roles here are specifically those in the DSC and are independent of any defined on the platform.

The ADM-R role provides sufficient privileges to manage the functionality of the DSC. As a role designed for the administrator, this role may be responsible for the following:

- Manage access control for SDOs (does not mean the contents of any particular SDO can be read by the administrator)
- Manage the configuration of the DSC

The MFGADM-R role, if explicitly defined in a DSC, may include the following responsibilities (which may be part of the ADM-R role otherwise):

- Manage the pre-installed SDOs and configuration of the DSC

The CApp-R role is focused on utilization of the functionality provided by the DSC. The following

would be representative of the responsibilities for this role:

- Requesting the creation of the SDOs
- Accessing or modifying created SDOs
- Deleting created SDOs

The ADM-R role does not mean that an administrator may be able to read the contents of any SDO even though it may be able to manage access rights on the SDO.

The management of timely updates (security or functional) for the DSC may be handled in a variety of ways, and as such may be associated with any role (though clearly this should be restricted to a well-defined "user").

In general, the CApp-R role is expected to be the primary role used when the components of the platform call to access the services provided by the DSC. Some examples of entities that may call the DSC and utilize the CApp-R role include:

- A content provider controlling access to its content through an application.
- A human entity using the platform who has an identity that they use to authenticate themselves to the content provider through a CApp.
- An application vendor acting on its own behalf to update software on the platform.
- An original equipment manufacturer (OEM) that designed and manufactured a more complex system with the DSC as a component (assuming that the DSC manufacturer and the manufacturer of the more complex system using the DSC as a component are different entities).

2.3. TOE Use Cases

DSCs are used in platforms to support mobile commerce, to manage platform credentials, manage user access to sensitive resources such as enterprise data centers or entertainment content servers, to manage and protect data-in-transit such as through secure channels or VPN tunnels, and to manage and protect keying, authentication, and authorization material for data-at-rest solutions such as self-encrypting drives.

For the mobile commerce use case, users, merchants, and financial institutions expect and require that financial transactions between them and their platforms be trusted and secure. For example,

- All peers to a transaction must be able to authenticate each other.
- The integrity of the transaction must be ensured.

To support such transactions, a DSC performs the following:

- Ingests data elements and attributes and exports the data objects associated with these transactions and the identities of the parties
- Generates data objects to use for these transactions.
- Securely stores data elements bound with their attributes within a protected hardware boundary.

- Authenticates and processes these data elements within a protected execution environment to ensure the authenticity of the parties and the transactions.
- Establishes secure communications channels between the parties to ensure the integrity and confidentiality of the transactions.
- Securely erases data objects when no longer needed.
- Ensures its own integrity and authenticity prior to execution.

DSCs are implemented to satisfy the following use cases:

[USE CASE 1] Protected Key Store

A platform leveraging DSCs as a hardware-secured Private Key Store facilitates the use of secure and protected storage of secret symmetric keys and private asymmetric keys for access to data and services. These DSCs would provide safe use of the private and secret keys inside the protected hardware boundary.

[USE CASE 2] User / Platform Authentication to Enterprise Managed Resources

A platform leveraging DSCs for a hardware-secured ID facilitates the use of the platform as a secure and reliable form of authentication for authorized access to highly sensitive local or remote data and services.

[USE CASE 3] Mobile Commerce

A platform that uses DSCs facilitates secure storage and protected use of credentials for financial transactions between trusted and authorized users, platforms, merchants and financial institutions. These DSCs would provide safe use of the credentials inside the protected hardware boundary. The use of certified hardware-isolated credential stores on smart platforms and only unlocking their use with authenticated authorization provides confidence that the transaction was indeed authorized by the approved 'platform holder'.

2.4. Key Reference Model

The Key Reference Model abstraction draws inspiration from several different DSC products. The products distinguish themselves from one another in the types of keys supported, how they are protected, the types of applications supported, the number of layers of key, and the number of keys at each layer.

The following paragraphs describe the relationships between elements of the DSC.

2.4.1. Key Usage

One way to categorize keys is by the cryptographic functions they are allowed to participate in. When one creates a key, one often restricts its use to encryption and decryption, or to signature generation and verification. There are exceptions to this rule, especially in proof of possession protocols. However, certification regimes often require strict separation of usage in regards to encryption/decryption and signature generation/verification: one may use a key for one or the other, but never both. As such, a DSC may have to enforce this separation of usage for keys; this

may mean that an attribute must accompany a key to help the DSC in its enforcement.

2.4.2. Sessions

For a DSC, a connection is established between a CApp and the services provided by the DSC when keys or services are requested. Each time the CApp establishes a connection to the DSC the CApp is authenticated to ensure the CApp has authorization to the requested keys. Since a CApp (acting as an agent of the user) may utilize their DSC keys multiple times, the establishment of individual connections for each use can be a resource constraint for the DSC as authorization methods using public keys tend to be resource intensive (i.e. uses a fair amount of internal memory and takes a long time).

As an alternative to requiring authorization for each access to a key, the DSC could allow the user or owner of the key to open a session. With a session, the CApp would provide the authentication data for the first connection, then the DSC would maintain the session and authorization using a series of less resource-intensive challenges and responses. In some instances the DSC may still require additional authorization (such as an elevation of privileges) to access keys (or different, related keys). Such a protocol of challenges and responses may generate and use ephemeral authorization tokens, which would be one form of critical security parameter (CSP). The DSC may have to switch session contexts in and out of the DSC to external temporary storage, which necessitates the protection of these CSPs. Such a session context is one type of SDO.

A session is a local connection only, between the CApp on the platform containing the DSC and the DSC itself. If the activities involving the DSC involve a connection with a remote system (i.e. something not on the platform), such a remote connection is the responsibility of the CApp. While the DSC may assist in establishing the remote connection, the DSC itself is only aware of the session between the CApp and the DSC itself.

Where a DSC may support a direct connection to a remote entity, this connection is established over a channel, with its own separate requirements (a channel may also support sessions, but that is dependent on the protocol used).

2.4.3. Key Hierarchies

Another way to categorize keys is the relationship they have with each other. A DSC may have a key hierarchy, or key chain, whereby data at rest is protected by one or more keys, which are protected in turn by one or more additional keys, and potentially so on. This model calls out three categories of keys generally found on typical DSCs. DSCs may contain Root Keys, Intermediate (or Branch) Keys, and Leaf Keys.

Most DSCs have a concept of Root Keys. These keys are typically provisioned by the DSC manufacturer and have some permanence in the DSC. Root Keys may be derived from seeds (which is discussed later), injected at manufacturing time, or provisioned by a user. Root keys installed by the manufacturers are considered administrator key material. Typically, normal client applications, including OEMs, should not alter or erase this material unless specifically authorized to do so. Root keys installed by the administrator should be similarly restricted. Client application-installed root keys, on the other hand, are not considered as permanent since the client application or the administrator can remove them at any time without authorization.

Root Keys may either be encryption/decryption keys, signature verification keys, or signature generation keys. Encryption/decryption keys, or simply Root Encryption Key (REK), usually anchor a hierarchy of keys stored external to the DSC necessitating both the encryption key to protect the key outside the DSC, and the decryption key to expose its contents within the protected and secure confines of the DSC. The signature verification keys from public key schemes should always contain the public portion and never the private portion. Use of signature generation keys as Root Keys is rare.

Most DSCs have a concept of Intermediate Keys. These are sometimes known as Branch Keys, Key Encryption Keys, and Key Wrapping Keys. In the SFRs of this cPP, these will be referred to as Key Encryption Keys (KEKs), even if the target of encryption is not a key. Intermediate Keys must always be encryption/decryption keys. Intermediate Keys cannot be signing keys.

Note that although chained certificates (see certificates below) are one form of a sequence of keys, each of which signs another key, the creation and verification of such a chain of certificates is out of scope for the core requirements of the cPP; however, it may be added as a package if one or both of these features (creating the chain and verifying the chain) is indeed present in the DSC. Nonetheless, the primitives of signing and verification are present due to other cryptographic operations in scope for this cPP.

Intermediate Keys should always be protected (i.e. wrapped) by either a Root Key or another Intermediate Key.

Leaf Objects consist of Authorization Data and Leaf Keys. Leaf Keys can be either encryption, decryption, signature generation, or signature verification keys. Leaf Objects collectively refers to data that should be wrapped by either a Root Key or a KEK and is not subsequently used as a KEK itself. Leaf Keys used for encryption/decryption do not wrap other keys (at least in the context of the DSC; what happens outside the DSC with Leaf Keys is out of its control). In many contexts, a Leaf Key used for encryption/decryption is known as a Data Encryption Key (DEK). In the context of the DSC, this cPP will not assume how the user of the DSC will use the Leaf Keys it creates, and will refrain from using the term DEK.

Certificates contain either signed public keys or some sort of Authorization Data. Signature keys come in several varieties: signature generation keys, which contain a private key for signing (and maybe also the public key for verification) and signature verification keys, which contain only the public verification key and do not contain the private key (and thus cannot perform a signing function). There are also symmetric signature keys. In this case these consist of only a single key for both signing and verifying.

Authorization Data may have an arbitrary length of bits or bytes and may contain arbitrary or non-arbitrary values of bits or bytes.

Seeds have a special place in this Key Reference Model. Manufacturers, owners, and users of the DSC can use permanent seeds to create Root Keys. Manufacturers have good reasons to use seeds to derive Root Keys and other items in the Key Reference Model. These include:

- Seeds take less space to store than certain asymmetric keys for given desired cryptographic strengths.
- Having seeds that are unique per DSC increases the probability that the same key derivation

function on different DSCs will yield unique keys.

Figure 5 contains an example of a hierarchy of keys where each lower-level key is wrapped by a higher-level key that is connected to it. The Root Encryption Key is an example of a Root Key. The numbered Key Encryption Keys are examples of Intermediate Keys. The Data Encryption Keys and Stored Keys are examples of Leaf Objects. Figure 5 serves as an illustration of key hierarchies; other configurations are possible.

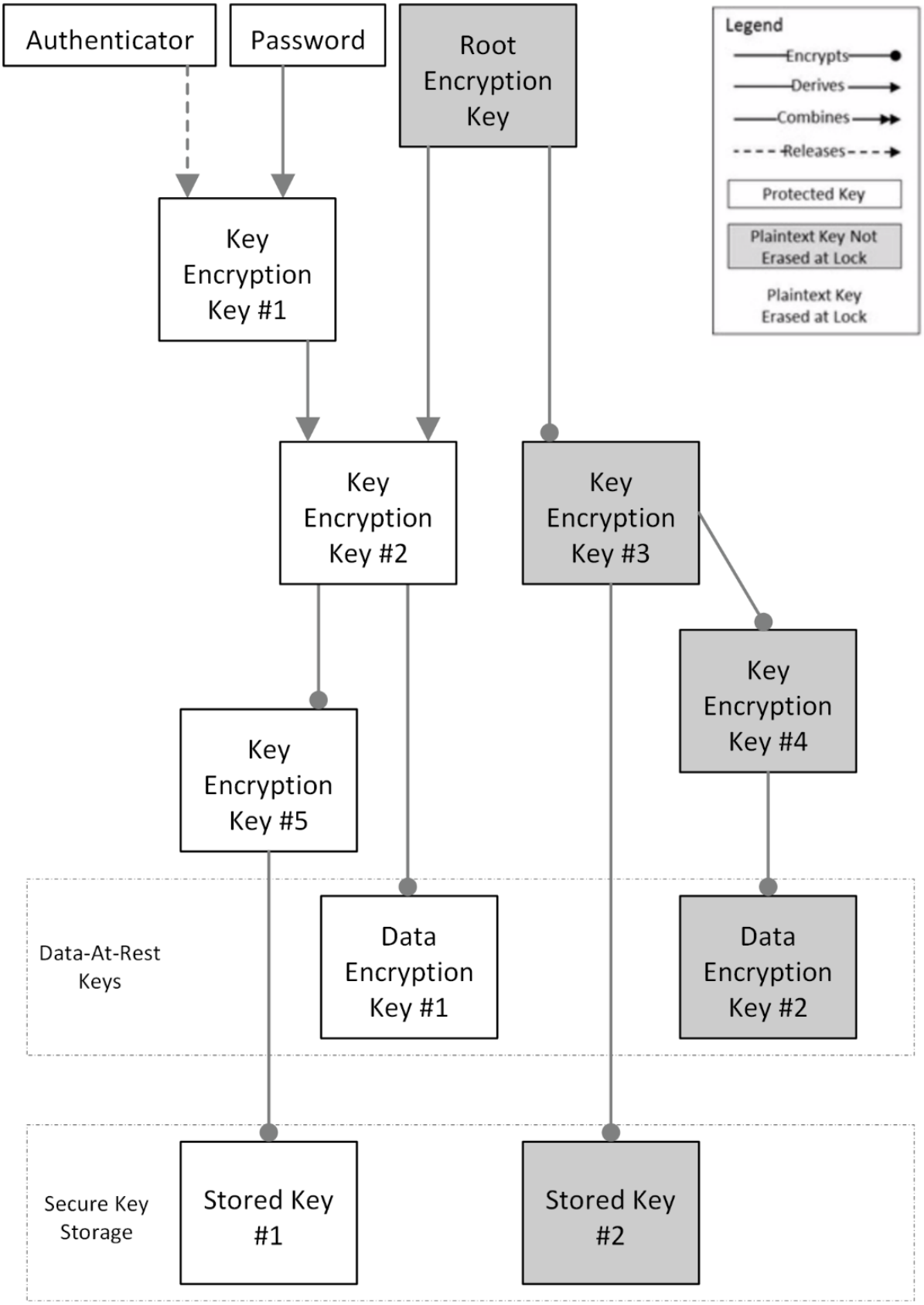


Figure 5. Example Key Hierarchy

Roles may play an important part in key hierarchies. One of the simplest models enforces a different hierarchy for each role at the Root Key level. Another way to put this is each hierarchy at the Root Key level supports a different role. However, for more complexity, once Intermediate Keys are allowed, then each Intermediate Key could serve as the root of a hierarchy of keys for a different role. Here is where the key functions and the roles come together. Roles may further

divide into which role has the right to use a key, which role has the right to move the key from one parent to another, which role has the right to destroy a key, etc.

2.4.4. Protected Storage Locations

This cPP covers several different types of storage locations for keys and critical security parameters (CSPs) such as authentication tokens. Some DSCs may have a generous amount of protected storage internal to themselves, which allows it to accommodate all keys and CSPs in operational use, whether the DSC is performing operations to administer itself or operations on behalf of users. Other DSCs may have a minimal amount of protected storage locations with just enough to accommodate root keys along with a limited number of operational keys and CSPs for user authorized sessions.

For those cases in which the DSC relies on storage external to itself to accommodate all the keys and CSPs on which applications expect it to operate, it will either have to support secure channels to another DSC with a more generous allocation of protected storage locations, or use a series of wrapping keys to protect private keys and CSPs while outside of the DSC. Whether the DSC is powered on or powered off, the DSC is expected to provide support for protected storage locations for its Root Keys. If the DSC uses external storage without secure channels, then it should be ready to wrap both Intermediate Keys as well as the Leaf Objects. This implies that there will be some sort of structure on each of these items stored external to the DSC. The next section discusses that structure.

A conformant TOE may include "write-once" storage such as single-use eFuses. Since data is written to any such storage as part of the initial provisioning of the TOE, the data is considered immutable once the TOE has entered its evaluated configuration. The integrity of this data is maintained through the physical properties of its storage medium.

2.4.5. SDEs and SDOs

This section is used to map keys and authentication tokens to SDEs and SDOs. This cPP does not impose a strict structure on the items in the key hierarchy. An X.509 certificate is one example of a strict structure of a key with attributes. Collecting attributes of an SDE and composing an SDO structure with an SDE and attribute fields imposes temporal and storage penalties in all cases. In certain resource-constrained cases the attributes could be implicit.

In the previous section on protected storage locations, a DSC may have to use storage external to itself. In these cases, an SDO of a wrapped key may contain a number of important attributes, such as a pointer to its parent, authorization values, and other indications of the functions allowed (encrypt vs. sign). Alternatively, some or all attributes may be implied, which means that only the keys or CSPs themselves exist outside the DSC. In either case, the sensitive values, such as private keys, secret keys, and CSPs, should be encrypted when outside the DSC. The parents of these objects are either Intermediate Keys, or encrypting Root Keys.

Some DSCs may want to distinguish between SDEs created within itself from SDEs ingested from an external source. Additionally, some DSCs may output SDEs without additional context or attributes from the DSC. A DSC, in some contexts, will not distinguish an ingested SDE from raw keys.

Chapter 3. CC Conformance Claims

As defined by the references [CC1], [CC2], [CC3], [CC4], [CC5] and [CC-E&I], this cPP:

- conforms to the requirements of Common Criteria CC:2022, Release 1 and the Errata and Interpretation, Version 1.1
- is Part 2 extended, Part 3 conformant
- does not claim conformance to any other PP or package.

The methodology applied for the cPP evaluation is defined in [CEM] and refined by the Evaluation Activities in [DSC SD]. This cPP satisfies the following Assurance Families: APE_CCL.1, APE_ECD.1, APE_INT.1, APE_OBJ.1, APE_REQ.1 and APE_SPD.1.

In order to be conformant to this cPP, a TOE must demonstrate Exact Conformance. Exact Conformance is defined as the ST containing all of the requirements in Section 6 of this cPP (these are the mandatory SFRs), and potentially requirements from Appendix A (these are optional SFRs) or Appendix B (these are selection-based SFRs, some of which will be mandatory according to the selections made in other SFRs) of this cPP. While iteration is allowed, no additional requirements (from CC Parts 2 or 3, or definitions of extended components not already included in this cPP) are allowed to be included in the ST. Further, no requirements in Section 6 of this cPP are allowed to be omitted.

The PPs and PP-Modules that are allowed to be specified in a PP-Configuration with this cPP are specified on the DSC-iTC website Allowed Components page.

Chapter 4. Security Problem Definition

4.1. Threats

T.BRUTE_FORCE_AUTH: An unauthorized user may attempt to gain unauthorized access to the TOE by repeatedly and rapidly supplying a large number of permutations of authorization data, such as passwords, biometrics, etc. that protect the SDEs, in the hopes that valid authorization data can be obtained through brute force.

T.HW_ATTACK: An individual with physical access to the TOE may apply hardware attacks such as probing, physical manipulation, fault injection, environmental stress, or reactivating blocked test-features or other pre-delivery services to manipulate the behavior of the TOE to disclose SDEs.

T.SDE_TRANSIT_COMPROMISE: An attacker with the ability to observe data transmission into and out of the TOE may access or determine plaintext values of keys, authorization data, and other SDEs as the TSF transmits them into or out of the TOE.

T.UNAUTH_UPDATE: An unauthorized user may force the platform to update the TOE with firmware that compromises its security features. Poorly chosen update protocols, cryptographic algorithms, and keys sizes may allow adversaries to install software or firmware that bypasses security features or rolls back to firmware versions with compromised security features and provides them with unauthorized access to SDEs.

T.UNAUTHORIZED_ACCESS: An unauthorized user may gain unauthorized access to one or more SDEs within the TOE. If an adversary gains access to SDEs stored in the TSF, they may attempt to view, use, or destroy this data as well as impersonate a user or that user's platform.

T.WEAK_CRYPTO: An unauthorized user or attacker that observes network traffic transmitted to and from the TOE may cryptographically exploit poorly chosen cryptographic algorithms, random bit generators, ciphers or key sizes. Weak cryptography chosen by users or by TSF protection mechanisms puts the user's data (including SDEs), identity, and platform at risk of exploitation by adversaries.

T.WEAK_ELEMENT_BINDING: An unauthorized user may successfully break the association between SDEs, for example to replace one element with another element.

T.WEAK_OWNERSHIP_BINDING: A user may successfully access or manipulate SDEs that they do not own.

4.2. Organizational Security Policies

There are no organizational security policies defined in this cPP.

4.3. Assumptions

This section describes the assumptions made in identification of the threats and security requirements for dedicated security components. The dedicated security component is not expected to provide assurance in any of these areas, and as a result, requirements are not included

to mitigate the threats associated.

A.AUTH_USERS: Authorized users follow all provided guidance regarding the safeguarding of SDEs held outside the TOE.

A.CREDENTIAL_REVOCATION: If a platform is lost, stolen, or compromised then there is a method of revocation of any credentials held (or equivalent method of mitigating the impact of potential access to the credentials). Credential revocation ensures that the loss of physical custody does not have significant negative impact on the security of the platform. This implies that an attacker has only limited access to the device to apply attacks. It further implies that the device owner is not seen as an attacker.

A.ROT_INTEGRITY: The vendor provides a RoT that is comprised of the TOE firmware, hardware, and pre-installed SDOs, free of intentionally malicious capabilities. The platform trusts the RoT since it cannot verify the integrity and authenticity of the RoT. Trust in the RoT may be intrinsic in the case of an immmutable RoT, while a mutable RoT will verify the authenticity and integrity of the updates before applying them.

A.TRUSTED_PEER: The remote peer communicating over a secure channel is trustworthy, and will not abuse the secure channel in order to introduce malware or fraudulent SDEs into the TOE.

Chapter 5. Security Objectives

5.1. Security Objectives for the TOE

This cPP is a Direct Rationale PP following Appendix B.5 of CC:2022 Part 1. Accordingly, no security objectives for the TOE are defined.

5.2. Security Objectives for the Operational Environment

The Operational Environment of the TOE implements technical and procedural measures to assist the TOE in correctly providing its security functionality. This section defines security objectives for the Operational Environment and consists of a set of statements describing the goals that the Operational Environment should achieve.

OE.AUTH_USERS: Authenticated users follow all provided guidance regarding the safeguarding of SDEs, especially authentication tokens such as passwords, pass-phrases, and biometrics.

OE.PHYSICAL: The platform holder will ensure that an attacker has no prolonged, unsupervised physical access to the platform. If a platform is lost or stolen then the platform holder will promptly initiate revocation of any credentials held (or equivalent method of mitigating the impact of potential access to the credentials). The platform may initiate the revocation based on local conditions or in response to remote signals such as from a service provider on the request of the platform holder.

OE.TRUSTED_PEER: Connections using secure channels are made only to trusted peers, in whom confidence has been established that they will not abuse the secure channel in order to introduce malware or fraudulent SDEs into the TOE.

5.3. Security Objectives Rationale

Table 3 shows the mapping of Security Objectives for the Operational Environment to Threats and Assumptions, along with rationale for these mappings. This mapping is provided in compliance with CC:2022 Part 1 Appendix B.5.

Table 3. Security Problem Definition Mapping to Security Objectives

Objective	Threat or Assumption	Rationale
OE.AUTH_USERS	A.AUTH_USERS	This objective holds that sufficiently trained and trusted users will follow instructions as assumed.

Objective	Threat or Assumption	Rationale
OE.PHYSICAL	A.CREDENTIAL_REVOCATION	This objective ensures that an adversary will not have sufficient access to the TOE to exploit the login mechanism if the assumption holds that credential revocation is enforced upon a lost or stolen TOE.
	T.HW_ATTACK	This objective ensures that the adversary has only a limited window of opportunity to engage in a hardware attack on the physical TOE.
OE.TRUSTED_PEER	A.TRUSTED_PEER	This objective holds that if the TOE's Operational Environment is configured such that the TSF can only communicate with trusted peer, then this assumption will be satisfied.
	A.ROT_INTEGRITY	This objective holds that the vendor's RoT can be relied upon if the only entities that the TSF communicates with are trusted.

The objectives can map to multiple assumptions or threats to fully define the objectives of the TOE and the operational environment.

Chapter 6. Security Functional Requirements

The individual security functional requirements are specified in the sections below. Based on selections made in these SFRs it will also be necessary to include some of the selection-based SFRs in Appendix B. Additional optional SFRs may also be adopted from those listed in Appendix A for those functions that are provided by the TOE instead of its Operational Environment.

The Evaluation Activities defined in [DSC SD] describe actions that the evaluator shall take in order to determine compliance of a particular TOE with the SFRs. The content of these Evaluation Activities will therefore provide more insight into deliverables required from TOE Developers.

6.1. Conventions

The conventions used in descriptions of the SFRs are as follows:

- Unaltered SFRs are stated in the form used in [CC2] or their extended component definition (ECD);
- Refinement made in the PP: the added/removed text is indicated with **bold text**/~~strikethroughs~~. When text is substituted (i.e. some text is added in place of some other text, which is then deleted), only the added text is included;
- Selections wholly or partially completed in the PP: the selection values (i.e. the selection values adopted in the PP or the remaining selection values available for the ST) are indicated with underlined text;

  For example, "[selection: *disclosure, modification, loss of use*]" in [CC2] or an ECD might become "[disclosure]" (completion) or "[selection: disclosure, modification]" (partial completion) in the PP;

- Assignment wholly or partially completed in the PP: indicated with *italicized text*;
- Assignment completed within a selection in the PP: the completed assignment text is indicated with *italicized and underlined text*

  For example, "[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) or "[selection: change default, select tag, [*select value*]]" (partial completion of selection, and completion of assignment) in the PP;

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

If the original SFR defined an assignment operation which was completed by the PP author by redefining it as a selection operation, this is considered to be a refinement of the original SFR, and the word "selection" will be indicated with **bold text**. The selection options will be indicated with *italicized text* to indicate the completion of the original assignment.

If the selection or assignment is to be completed by the ST author, it is preceded by 'selection:' or 'assignment:'. If the selection or assignment has been completed by the PP author and the ST author

does not have the ability to modify it, the proper formatting convention is applied but the preceding word is not included. The exception to this is if the SFR definition includes multiple options in a selection or assignment and the PP has excluded certain options but at least two remain. In this case, the selection or assignment operations that are not permitted by this PP are removed without applying additional formatting and the 'selection:' or 'assignment:' text is preserved to show that the ST author still has the ability to choose from the reduced set of options.

Some 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. Individual entries in these tables may also require further selections or assignments. Within the tables, the selections and assignments just follow the normal conventions as the specific modifications applied to the SFR are included in the SFR itself, and the table will only follow the normal conventions under that specified within the SFR.

For example, for the Table 4, the ST for a TOE that supports RSA key pair generation must include the entries for 'Cryptographic Key Generation Algorithm', 'Cryptographic Algorithm Parameters', and 'List of Standards'. For 'Cryptographic Algorithm Parameters', the ST author must further select which of the required parameter information are supported. Likewise, if the TOE supports ECC key pair generation, the ST must include the entries from the appropriate ECC row along with the appropriate selections.

Table 4. Sample Cryptographic Table

Cryptographic Key Generation Algorithm	**Cryptographic Algorithm Parameters**	**List of Standards**
RSA	Modulus of size [selection: 2048 bit, 3072 bit]	NIST FIPS PUB 186-5 (Section A.1.1)
ECC - Extra Random Bits	Elliptic Curve [selection: P-256, brainpoolP256r1, P-384, brainpoolP384r1, P-521, brainpoolP512r1]	[selection: NIST FIPS PUB 186-5 (Section A.2.1), NIST SP 800-56A Rev. 3 (Section 5.6.1.2.1)] [selection: NIST SP 800-186 (Section 4) [NIST Curves], RFC 5639 (section 3) [brainpool curves]]
ECC - Rejection Sampling	Elliptic Curve [selection: P-256, brainpoolP256r1, P-384, brainpoolP384r1, P-521, brainpoolP512r1]	[selection: NIST FIPS PUB 186-5 (Section A.2.2), NIST SP 800-56A Rev. 3 (Section 5.6.1.2.2)] [selection: NIST SP 800-186 (Section 4) [NIST Curves], RFC 5639 (section 3) [brainpool curves]]

Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
FFC - Extra Random Bits	Static domain parameters approved for [selection: IKE groups [selection: MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [selection: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]	NIST SP 800-56A Rev. 3, RFC 3526, RFC 7919 [FFC domain parameters] NIST SP 800-56A Rev. 3 (Section 5.6.1.1.3) [key pair generation]

Extended SFRs (i.e. those SFRs that are not defined in [CC2]) are identified by having a label '_EXT' at the end of the SFR name.

6.2. Cryptographic Support

The DSC ensures that the cryptography used is commensurate with the SDEs and SDOs that are being protected based on the selections made in the SFRs. The shortest key strength determines the overall security strength for the DSC.

6.2.1. FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1

The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **corresponding to [selection**:

- *Asymmetric keys generated in accordance with FCS_CKM.1/AKG,*
- *Symmetric keys generated in accordance with FCS_CKM.1/SKG,*
- *Derived keys generated in accordance with FCS_CKM.5*
- *Derived keys generated in accordance with FCS_CKM_EXT.8*

] ~~and specified cryptographic key sizes [*assignment: cryptographic key sizes*] that meet the following: [*assignment: list of standards*]~~.

Application Note 1

Cryptographic keys can include KEKs that protect keys as well as the keys used to protect SDEs and SDOs.

6.2.2. FCS_CKM.2 Cryptographic Key Distribution

FCS_CKM.2 Cryptographic Key Distribution

FCS_CKM.2.1

The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method [**selection**: *key encapsulation as specified in FCS_COP.1/KeyEncap, key*

wrapping as specified in FCS_COP.1/KeyWrap, physically protected channels as specified in FTP_ITP_EXT.1, encrypted data buffers as specified in FTP_ITE_EXT.1, cryptographically protected data channels as specified in FTP_ITC_EXT.1] that meets the following: [*none*].

Application Note 2

This SFR is for cases where there are no pre-shared key between the parties.

6.2.3. FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM.6.1

The TSF shall destroy [assignment: *list of cryptographic keys (including keying material)*] when [selection: *no longer needed, [assignment: other circumstances for key or keying material destruction]*].

Application Note 3

The TOE will have mechanisms to destroy keys, including intermediate keys and key material, by using an approved method as specified in FCS_CKM.6.2. Examples of keys include intermediate keys, leaf keys, encryption keys, and signing keys. Key material includes seeds, authentication secrets, passwords, PINs, and other secret values used to derive keys. All such keys and keying material are subject to destruction in the first assignment.

This SFR does not apply to the public component of asymmetric key pairs or to keys that are permitted to remain stored, such as device identification keys.

FCS_CKM.6.2

The TSF shall destroy cryptographic keys and keying material specified by FCS_CKM.6.1 in accordance with a specified cryptographic key destruction method [**selection**:

1. *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 CSP]],*
    b. *removal of power to the memory,*
    c. *removal of all references to the key directly followed by a request for garbage collection];*
2. *For non-volatile memory [selection:*
    a. *that employs a wear-leveling algorithm, the destruction shall be executed by a [selection:*
        i. *single overwrite consisting of [selection: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [assignment: some value that does not contain any CSP]],*
        ii. *block erase];*

b. *that does not employ a wear-leveling algorithm, the destruction shall be executed by a [selection:*

   i. *[selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of [selection: zeros, ones, pseudo-random pattern, a new value of a key of the same size, [assignment: some value that does not contain any CSP]] followed by a read-verify. If the read-verification of the overwritten data fails, the process shall be repeated again up to [assignment: number of times to attempt overwrite] times, whereupon an error is returned.*

   ii. *block erase]*

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

Application Note 4

In the case of volatile memory, the selection "removal of all references 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) and then requesting the platform 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 selection for destruction of data in non-volatile memory includes block erase as an option, and this option applies only to flash memory. A block erase does not require a read verify, since the mappings of logical addresses to the erased memory locations are erased as well as the data itself.

Some selections allow assignment of "some value that does not contain any CSP." This means that the TOE uses some specified data not drawn from an RBG meeting FCS_RBG requirements, and not being any of the particular values listed as other selection options. The point of the phrase "does not contain any sensitive data" is to ensure that the overwritten data is carefully selected, and not taken from a general pool that might contain data that itself requires confidentiality protection.

6.2.4. FCS_COP.1/Hash Cryptographic Operation - Hashing

FCS_COP.1/Hash Cryptographic Operation - Hashing

FCS_COP.1.1/Hash

The TSF shall perform [*cryptographic hashing*] in accordance with a specified cryptographic algorithm [**selection**: *SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512*] that meets the following: [**selection**: *ISO/IEC 10118-3:2018 [SHA, SHA3], FIPS PUB 180-4 [SHA], FIPS PUB 202 [SHA3]*].

Application Note 5

SHA-1 is allowed in cases where collision resistance is not required. 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. SHA-1 may also be used for verifying old digital signatures and time stamps, if this is explicitly allowed by the application domain.

6.2.5. FCS_COP.1/KeyedHash Cryptographic Operation - Keyed Hash

FCS_COP.1/KeyedHash Cryptographic Operation - Keyed Hash

FCS_COP.1.1/KeyedHash

The TSF shall perform [*keyed hash message authentication*] in accordance with a specified cryptographic algorithm [**selection**: *keyed hash algorithm*] and cryptographic key sizes [**selection**: *cryptographic key size*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/KeyedHash.

Table 5. Allowed choices for FCS_COP.1/KeyedHash

Keyed Hash Algorithm	Cryptographic Key Sizes	List of Standards
HMAC-SHA-1	[selection: (ISO, FIPS) 160, (FIPS) 128] bits	[selection: ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"); FIPS PUB 198-1]
HMAC-SHA-224	[selection: (ISO, FIPS) 224, (FIPS) 192, 128] bits	[selection: ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"); FIPS PUB 198-1]
HMAC-SHA-256	[selection: (ISO, FIPS) 256, (FIPS) 192, 128] bits	[selection: ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"); FIPS PUB 198-1]
HMAC-SHA-384	[selection: (ISO, FIPS) 384, (FIPS) 256, 192, 128] bits	[selection: ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"); FIPS PUB 198-1]
HMAC-SHA-512	[selection: (ISO, FIPS) 512, (FIPS) 384, 256, 192, 128] bits	[selection: ISO/IEC 9797-2:2021 (Section 7 "MAC Algorithm 2"); FIPS PUB 198-1]
KMAC128	128 bits	[selection: ISO/IEC 9797-2:2021 (Section 9 "MAC Algorithm 4"); NIST SP 800-185 (Section 4 "KMAC")]
KMAC256	256 bits	[selection: ISO/IEC 9797-2:2021 (Section 9 "MAC Algorithm 4"); NIST SP 800-185 (Section 4 "KMAC")]
KMACXOF128	[assignment: integer $256 \Leftarrow Lk < 2^{2040}$]	[selection: ISO/IEC 9797-2:2021 (Section 9 "MAC Algorithm 4"); NIST SP 800-185 (Section 4 "KMAC")]
KMACXOF256	[assignment: integer $256 \Leftarrow Lk < 2^{2040}$]	[selection: ISO/IEC 9797-2:2021 (Section 9 "MAC Algorithm 4"); NIST SP 800-185 (Section 4 "KMAC")]

Application Note 6

The HMAC minimum key sizes in the table are specified in ISO/IEC 9797-2:2021, which requires that

the minimum key size be equal to the digest size. The FIPS standard specifies no minimum or maximum key sizes, so if FIPS PUB 198-1 is selected, larger or smaller key sizes may be used. This is indicted by the parenthesized annotations in the Cryptographic Key Sizes column.

6.2.6. FCS_COP.1/SigGen Cryptographic Operation - Signature Generation

FCS_COP.1/SigGen Cryptographic Operation - Signature Generation

FCS_COP.1.1/SigGen

The TSF shall perform [*digital signature generation*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic **algorithm parameters** [**selection**: *cryptographic algorithm parameters*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/SigGen.

Table 6. Allowed choices for FCS_COP.1/SigGen

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [selection: 2048, 3072, 4096] bits, hash or XOF [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [selection: 2048, 3072, 4096] bits, hash or XOF [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512, SHAKE128 with 256-bit output, SHAKE256 with 512-bit output]	RFC 8017 (Section 8.1) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
ECDSA	ECDSA	Elliptic Curve [selection: P-256, brainpoolP256r1, P-384, brainpoolP384r1, P-521, brainpoolP512r1], per-message secret number generation [selection: extra random bits, rejection sampling, deterministic] and hash or XOF function using [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512, SHAKE128 with 256-bit output, SHAKE256 with 512-bit output]	[selection: ISO/IEC 14888-3:2018 (Subclause 6.6), FIPS PUB 186-5 (Sections 6.3.1, 6.4.1)] [ECDSA] [selection: RFC 5639 (Section 3) [Brainpool Curves], NIST SP-800 186 (Section 4) [NIST Curves]]
KCDSA	KCDSA	hash function using [selection: SHA-224, SHA-256, SHA-384, SHA-512]	ISO/IEC 14888-3:2018 (Subclause 6.3) [KCDSA]
EC-KCDSA	EC-KCDSA	Elliptic Curve [selection: P-224, P-256, B-233, B-283, K-233, K-283] using hash [selection: SHA-224, SHA-256, SHA-384, SHA-512]	ISO/IEC 14888-3:2018 (Subclause 6.7) [EC-KCDSA] NIST SP 800-186 (Section 3) [NIST Curves]
EdDSA	Edwards-Curve Digital Signature Algorithm	Domain parameters approved for elliptic curves [selection: Edwards25519, Edwards448]	NIST FIPS PUB 186-5 (Section 7.6) [EdDSA] RFC 8032 [Edwards Curves]
LMS	LMS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], Winternitz parameter = [selection: 1, 2, 4, 8], and tree height = [selection: 5, 10, 15, 20, 25]	RFC 8554 [LMS] NIST SP 800-208 [parameters]

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
HSS	Multitree version of LMS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], Winternitz parameter = [selection: 1, 2, 4, 8], tree height = [selection: 5, 10, 15, 20, 25], and number of levels = [selection: 1, 2, 3, 4, 5, 6, 7, 8]	RFC 8554 [HSS] NIST SP 800-208 [parameters]
XMSS	XMSS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], tree height = [selection: 10, 16, 20]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]
XMSS^{MT}	Multitree version of XMSS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], (total tree height, number of levels) = [selection: (20, 2), (20, 4), (40, 2), (40, 4), (40, 8), (60, 3), (60, 6), (60, 12)]	RFC 8391 [XMSS^{MT}] NIST SP 800-208 [parameters]
ML-DSA	ML-DSA.Sign	Parameter set = [selection: ML-DSA-44, ML-DSA-65, ML-DSA-87]	NIST FIPS 204 (Section 5.2)
HashML-DSA	HashML-DSA.Sign	Parameter set = [selection: ML-DSA-44, ML-DSA-65, ML-DSA-87] and hash function [selection: SHA-256, SHA-512/256, SHA3-256, SHAKE128, SHA-384, SHA3-384, SHA-512, SHA3-512, SHAKE256]	NIST FIPS 204 (Section 5.4)

Application Note 7

The dependency on FCS_OTV_EXT.1 is needed only for signature schemes that require random bits,

such as ECDSA, KCDSA, or EC-KCDSA.

For LMS, HSS, XMSS, and XMSS^{MT}, the key sizes do not represent the expected security strength. All key sizes given here correspond to an expected security strength of 128 bits, per NIST SP 800-208.

For HSS and XMSS^{MT} the same hash or XOF function shall be used at each level. Within each level, the same Winternitz parameter shall be used but can be different for each level. For HSS, within each level, the same tree height shall be used but can be different for each level.

Per Section 5.4 of NIST FIPS 204, the collision strength of the hash function selected for the pre-hash of HashML-DSA must be at least the strength of the signature algorithm.

6.2.7. FCS_COP.1/SigVer Cryptographic Operation - Signature Verification

FCS_COP.1/SigVer Cryptographic Operation - Signature Verification

FCS_COP.1.1/SigVer

The TSF shall perform [*digital signature verification*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic **algorithm parameters** [**selection**: *cryptographic algorithm parameters*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/SigVer.

Table 7. Allowed choices for FCS_COP.1/SigVer

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
RSA-PKCS	RSASSA-PKCS1-v1_5	Modulus of size [selection: 2048, 3072, 4096] bits, hash or XOF [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512]	RFC 8017 (Section 8.2) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PKCS1-v1_5]
RSA-PSS	RSASSA-PSS	Modulus of size [selection: 2048, 3072, 4096] bits, hash or XOF [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512, SHAKE128 with 256-bit output, SHAKE256 with 512-bit output]	RFC 8017 (Section 8.1) [PKCS #1 v2.2] FIPS PUB 186-5 (Section 5.4) [RSASSA-PSS]
DSA	DSA	Domain parameters for (L, N) = [selection: (2048, 224), (2048, 256), (3072, 256)] bits	FIPS PUB 186-4 (Section 4.7) [DSA Signature Verification]

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
ECDSA	ECDSA	Elliptic Curve [selection: P-256, brainpoolP256r1, P-384, brainpoolP384r1, P-521, brainpoolP512r1] using hash or XOF function using [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512, SHAKE128 with 256-bit output, SHAKE256 with 512-bit output]	[selection: ISO/IEC 14888-3:2018 (Subclause 6.6), FIPS PUB 186-5 (Section 6.4.2)] [ECDSA] [selection: RFC 5639 (Section 3) [Brainpool Curves], NIST SP-800 186 (Section 4) [NIST Curves]]
KCDSA	KCDSA	hash function using [selection: SHA-224, SHA-256, SHA-384, SHA-512]	ISO/IEC 14888-3:2018 (Subclause 6.3) [KCDSA]
EC-KCDSA	EC-KCDSA	Elliptic Curve [selection: P-224, P-256, B-233, B-283, K-233, K-283] using hash [selection: SHA-224, SHA-256, SHA-384, SHA-512]	ISO/IEC 14888-3:2018 (Subclause 6.7) [EC-KCDSA] NIST SP 800-186 (Section 3) [NIST Curves]
EdDSA	Edwards-Curve Digital Signature Algorithm	Domain parameters approved for elliptic curves [selection: Edwards25519, Edwards448]	NIST FIPS PUB 186-5 (Section 7.7) [EdDSA] RFC 8032 [Edwards Curves]
LMS	LMS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], Winternitz parameter = [selection: 1, 2, 4, 8], and tree height = [selection: 5, 10, 15, 20, 25]	RFC 8554 [LMS] NIST SP 800-208 [parameters]

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
HSS	Multitree version of LMS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], Winternitz parameter = [selection: 1, 2, 4, 8], tree height = [selection: 5, 10, 15, 20, 25], and number of levels = [selection: 1, 2, 3, 4, 5, 6, 7, 8]	RFC 8554 [HSS] NIST SP 800-208 [parameters]
XMSS	XMSS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], tree height = [selection: 10, 16, 20]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]
$XMSS^{MT}$	Multitree version of XMSS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], (total tree height, number of levels) = [selection: (20, 2), (20, 4), (40, 2), (40, 4), (40, 8), (60, 3), (60, 6), (60, 12)]	RFC 8391 [$XMSS^{MT}$] NIST SP 800-208 [parameters]
ML-DSA	ML-DSA.Verify	Parameter set = [selection: ML-DSA-44, ML-DSA-65, ML-DSA-87]	NIST FIPS 204 (Section 5.3)
HashML-DSA	HashML-DSA.Verify	Parameter set = [selection: ML-DSA-44, ML-DSA-65, ML-DSA-87] and hash function [selection: SHA-256, SHA-512/256, SHA3-256, SHAKE128, SHA-384, SHA3-384, SHA-512, SHA3-512, SHAKE256]	NIST FIPS 204 (Section 5.4)

Application Note 8

DSA may only be used to verify signatures generated prior to the implementation date of FIPS 186-

5.

The TOE may contain a public key which is integrity protected (e.g., in hardware), in which case the FDP_ITC.1 and FDP_ITC.2 dependencies do not apply. In this case, no dependencies may be chosen.

For signature verifications, private keys are not necessary, so there are no dependencies required for generating or destroying cryptographic keys.

For LMS, HSS, XMSS, and XMSS^{MT}, the key sizes do not represent the expected security strength. All key sizes given here correspond to an expected security strength of 128 bits, per NIST SP 800-208.

For HSS and XMSS^{MT} the same hash or XOF function shall be used at each level. Within each level, the same Winternitz parameter shall be used but can be different for each level. For HSS, within each level, the same tree height shall be used but can be different for each level.

Per Section 5.4 of NIST FIPS 204, the collision strength of the hash function selected for the pre-hash of HashML-DSA must be at least the strength of the signature algorithm.

6.2.8. FCS_COP.1/SKC Cryptographic Operation - Symmetric-Key Cryptography

FCS_COP.1/SKC Cryptographic Operation - Symmetric-Key Cryptography

FCS_COP.1.1/SKC

The TSF shall perform [*symmetric-key encryption/decryption*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic key sizes [**selection**: *cryptographic key sizes*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/SKC.

Table 8. Allowed choices for FCS_COP.1/SKC

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CBC	AES in CBC mode with IVs generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	[selection: ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [selection: ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]
XTS-AES	AES in XTS mode with tweak values generated according to FCS_OTV_EXT.1	[selection: 256, 512] bits	[selection: ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [selection: IEEE Std. 1619-2018, NIST SP 800-38E] [XTS]

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CTR	AES in CTR mode with counter values generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	[selection: ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [selection: ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A] [CTR]
CAM-CBC	Camellia in CBC mode with IVs generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 18033-3:2010 (Subclause 5.3) [Camellia] [selection: ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]
CAM-CFB	Camellia in CFB mode with IVs generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 18033-3:2010 (Subclause 5.3) [Camellia] [selection: ISO/IEC 10116:2017 (Clause 8), NIST SP 800-38A] [CFB]
CAM-OFB	Camellia in OFB mode with IVs generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 18033-3:2010 (Subclause 5.3) [Camellia] [selection: ISO/IEC 10116:2017 (Clause 9), NIST SP 800-38A] [CFB]
XTS-CAM	Camellia in XTS mode with tweak values generated according to FCS_OTV_EXT.1	[selection: 256, 512] bits	ISO/IEC 18033-3:2010 (Subclause 5.3) [Camellia] [selection: IEEE Std. 1619-2018, NIST SP 800-38E] [XTS]
CAM-CTR	Camellia in CTR mode with counter values generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 18033-3:2010 (Subclause 5.3) [Camellia] [selection: ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A] [CTR]
SEED-CBC	SEED in CBC mode with IVs generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 5.4) [SEED] [selection: ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
SEED-CFB	SEED in CFB mode with IVs generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 5.4) [SEED] [selection: ISO/IEC 10116:2017 (Clause 8), NIST SP 800-38A] [CFB]
SEED-OFB	SEED in OFB mode with IVs generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 5.4) [SEED] [selection: ISO/IEC 10116:2017 (Clause 9), NIST SP 800-38A] [OFB]
SEED-CTR	SEED in CTR mode with counter values generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 5.4) [SEED] [selection: ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A] [CTR]
HIGHT-CBC	HIGHT in CBC mode with IVs generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 4.5) [HIGHT] [selection: ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]
HIGHT-CFB	HIGHT in CFB mode with IVs generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 4.5) [HIGHT] [selection: ISO/IEC 10116:2017 (Clause 8), NIST SP 800-38A] [CFB]
HIGHT-OFB	HIGHT in OFB mode with IVs generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 4.5) [HIGHT] [selection: ISO/IEC 10116:2017 (Clause 9), NIST SP 800-38A] [OFB]
HIGHT-CTR	HIGHT in CTR mode with counter values generated according to FCS_OTV_EXT.1	128 bits	ISO/IEC 18033-3:2010 (Subclause 4.5) [HIGHT] [selection: ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A] [CTR]

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
LEA-CBC	LEA in CBC mode with IVs generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 29192-2:2019 (Subclause 6.3) [LEA] [selection: ISO/IEC 10116:2017 (Clause 7), NIST SP 800-38A] [CBC]
LEA-CFB	LEA in CFB mode with IVs generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 29192-2:2019 (Subclause 6.3) [LEA] [selection: ISO/IEC 10116:2017 (Clause 8), NIST SP 800-38A] [CFB]
LEA-OFB	LEA in OFB mode with IVs generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 29192-2:2019 (Subclause 6.3) [LEA] [selection: ISO/IEC 10116:2017 (Clause 9), NIST SP 800-38A] [OFB]
LEA-CTR	LEA in CTR mode with counter values generated according to FCS_OTV_EXT.1	[selection: 128, 192, 256] bits	ISO/IEC 29192-2:2019 (Subclause 6.3) [LEA] [selection: ISO/IEC 10116:2017 (Clause 10), NIST SP 800-38A] [CTR]

6.2.9. FCS_RBG.1 Random Bit Generation (RBG)

FCS_RBG.1 Random Bit Generation (RBG)

FCS_RBG.1.1

The TSF shall perform deterministic random bit generation services using [**selection**: *DRBG algorithm*] in accordance with [**selection**: *list of standards*] after initialization with a seed.

The following table provides the recommended choices for completion of the selection operations of FCS_RBG.1.

Table 9. Allowed choices for FCS_RBG.1

Identifier	DRBG Algorithm	List of Standards
HASH_DRBG	Hash_DRBG with [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512]	[selection: ISO/IEC 18031: 2011 (Section C.2.2), NIST SP 800-90A Revision 1 Section 10.1.1] FIPS PUB 202 [SHA3]

Identifier	DRBG Algorithm	List of Standards
HMAC_DRBG	HMAC_DRBG with [selection: SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512]	[selection: ISO/IEC 18031: 2011 (Section C.2.3), NIST SP 800-90A Revision 1 Section 10.1.2] FIPS PUB 202 [SHA3]
CTR_DRBG	CTR_DRBG with [selection: AES-128, AES-192, AES-256, SEED-128, HIGHT-128, LEA-128, LEA-192, LEA-256]	[selection: ISO/IEC 18031: 2011 (Section C.3.2), NIST SP 800-90A Revision 1 Section 10.2.1]

FCS_RBG.1.2

The TSF shall use a [selection: *TSF **entropy** source [assignment: name of **entropy** source], TSF interface for obtaining entropy*] for initialization and reseeding.

FCS_RBG.1.3

The TSF shall update the **DRBG** state by [selection: *reseeding, uninstantiating and re-instantiating*] using a [selection: *TSF **entropy** source [assignment: name of **entropy** source], TSF interface for obtaining entropy [assignment: name of the interface]*] in the following situations: [selection:

- *never,*
- *on demand,*
- *on the condition: [assignment: condition],*
- *after [assignment: time as supplied according to FPT_STM_EXT.1]*]

in accordance with [assignment: *list of standards*].

Application Note 9

No rationale is acceptable for not satisfying one of these dependencies.

If a reseeding is selected in the first selection and something other than "never" is selected in the third selection of FCS_RBG.1.3, but reseeding is not feasible, the TSF will uninstantiate RBGs, rather than produce output that is of insufficient quality. The listed standards should specify the reseed interval and procedure for uninstantiating and reseeding.

"Uninstantiate" means that the internal state of the DRBG is no longer available for use.

In the third selection for FCS_RBG.1.3, "on demand" means that a TOE presents an interface to reseed as a TSFI (e.g., an API call). The interface causes the DRBG to reseed at the request of an authorized user, either with an internal source, an external source, or from input provided through the TSFI (e.g., the API call).

6.2.10. FCS_OTV_EXT.1 One-Time Value

FCS_OTV_EXT.1 One-Time Value

FCS_OTV_EXT.1.1

The TSF shall perform cryptographic one-time value generation for [**selection**: *algorithm or mode*] using the output of a [selection: *random bit generator as defined in FCS_RBG.1, deterministic OTV construction, [assignment: OTV construction method]*] and sizes of length that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_OTV_EXT.1.

Table 10. Allowed choices and guidance for FCS_OTV_EXT.1

Algorithm or Mode	List of Standards	Notes
HMAC	FIPS PUB 198-1, NIST SP 800-56C Revision 2	Depending on the use case, salts can be secret or known, randomly generated, or all zero. Secret IVs may be required, e.g., for key derivation. Refer to the relevant standards for your use case.
KMAC	NIST SP 800-185, NIST SP 800-56C Revision 2	Depending on the use case, salts can be secret or known, randomly generated, or all zero. Secret IVs may be required, e.g., for key derivation. Refer to the relevant standards for your use case.
KDF	NIST SP 800-108 Revision 1, NIST SP 800-135 Revision 1, ISO/IEC 11770-6:2016 (Subclause 7.3.2)	Salts and IVs as directed in use of HMAC, AES, and CAM cryptographic algorithms. Refer to the relevant standards.
PBKDF	NIST SP 800-132	Salts generated and used as directed in PBKDFs.
CTR	NIST SP 800-38A	"Initial Counter" (nonce) shall be non-repeating. No counter value shall be repeated across multiple messages with the same secret key.
CBC	NIST SP 800-38A Appendix C	Depending on the use case, IVs shall be unpredictable. Repeating IVs leak information about whether the first one or more blocks are shared between two messages, so IVs should be non-repeating in such situations. Refer to the relevant standards for your use case.

Algorithm or Mode	List of Standards	Notes
OFB	NIST SP 800-38A	IVs shall be non-repeating and shall not be generated by invoking the cipher on another IV. OFB may require the IV to be a nonce.
CFB	NIST SP 800-38A	IVs should be non-repeating as repeating IVs leak information about the first plaintext block and about common shared prefixes in messages.
XTS	NIST SP 800-38E, IEEE Std. 1619-2018	Tweak values shall be non-negative integers, assigned consecutively, and starting at an arbitrary non-negative integer (i.e., sequential nonces).
CMAC	NIST SP 800-38B	IV is all zeroes.
KW, KWP	NIST SP 800-38F	Depending on the use case, nonces may be required. Please reference the relevant standards for your use case.
CCM	NIST SP 800-38C	Nonces shall be non-repeating.
GCM	NIST SP 800-38D	For RBG-based IV construction (section 8.2.2) the number of invocations of GCM shall not exceed 2^32 for a given secret key.
RSA-OAEP	NIST SP 800-56B Revision 2	Mask for padding shall be randomly generated.

Application Note 10

TSFs frequently generate cryptographic one-time values, often non-secret, such as nonces, IVs, salts, and initial counters (sometimes called initial sequential nonces) using the output of an RBG specified in FCS_RBG.1. If the TSF is generating OTVs, then this SFR is used.

Salts help protect against dictionary and other precomputation attacks. Systems often prepend or append salts to passwords and other long-term, potentially guessable values to increase the size of a dictionary an attacker must build to attack it. Salts, once associated with a password, generally do not change for the life of that password. Salts should also be unique for each password and should not be reused. Therefore, systems should randomly generate salts with sufficient size such that the combined entropy of both the salt and the password meets the minimal key strength sizes of the chosen algorithms.

Nonces help protect against replay attacks in cryptographic authentication protocols and some encryption modes. A nonce should never repeat. Using a sequence of nonces with a counter embedded in the value will ensure a nonce will never repeat. In protocol sessions that require multiple nonces, using sequential nonces that increment for each message—the receiver can check for and accept only an increase in the nonce value to verify that the message has not been replayed. In some protocols, the initial sequential nonce needs only to be sent once at the beginning of the

session and the receiver can predict the remaining nonces in that session, which saves transmission bandwidth. Randomly generated nonces protect against attacks against sessions in which multiple keys are expected to be used. Therefore, nonces should be both randomly generated and never repeat. However, sequential nonces may be predictable. NIST provides additional guidance for the composition of a nonce in NIST SP 800-38c, NIST SP 800-56A Revision 3, NIST SP 800-56B Revision 2, NIST SP 800-63B, and NIST SP 800-90A Revision 1.

Initialization Vectors (IVs) help protect against attacks which depend on the reuse of static keys. Certain encryption modes often require IVs. They should be randomly generated in a nonpredictable way, cannot be sequential, and cannot repeat.

Each algorithm and mode have varying guidance on the lengths of the salts, nonces, and initialization vectors used therein. Please consult the referenced standards documents for the appropriate guidance for each.

6.2.11. FCS_STG_EXT.1 Protected Storage

FCS_STG_EXT.1 Protected Storage

FCS_STG_EXT.1.1

The TSF shall provide [**selection**: *mutable hardware-based, immutable hardware-based, software-based in accordance to FCS_CKM_EXT.3*] protected storage for asymmetric private keys, symmetric keys and [selection: *persistent secrets, no other keys*].

FCS_STG_EXT.1.2

The TSF shall support the capability of [selection: *importing keys/secrets into the TOE, causing the TOE to generate keys/secrets*] upon request of [**selection**: *a client application, an administrator*].

FCS_STG_EXT.1.3

The TSF shall be capable of destroying keys/secrets in the protected storage upon request of [**selection**: *a client application, an administrator*].

FCS_STG_EXT.1.4

The TSF shall have the capability to allow only the user that [selection: *imported the key/secret, caused the key/secret to be generated*] to use the key/secret. Exceptions may only be explicitly authorized by [**selection**: *the client application, the administrator*].

FCS_STG_EXT.1.5

The TSF shall allow only the user that [selection: *imported the key/secret, caused the key/secret to be generated*] to request that the key/secret be destroyed. Exceptions may only be explicitly authorized by [**selection**: *the client application, the administrator*].

Application Note 11

Not all conformant TOEs will have the ability to import pre-generated keys into the TOE. In these cases, the TOE's ability to receive commands to perform key generation is considered to be its implementation of the Parse service. A subject that caused a key to be generated is considered to be the 'owner' of that key in the same manner as they would be if they had imported it directly.

6.3. User Data Protection

6.3.1. FDP_ACC.1 Subset Access Control

FDP_ACC.1 Subset Access Control

FDP_ACC.1.1

The TSF shall enforce the [*Access Control SFP*] on [

- *Subjects: S.DSC, S.Admin, S.CApp, S.EPS*
- *Objects: OB.P_SDO, OB.T_SDO, OB.AuthData, OB.Pstate, OB.FAACntr, OB.AntiReplay, OB.Context*
- *Operations: OP.Import, OP.Create, OP.Use, OP.Modify, OP.Attest, OP.Store, OP.Export, OP.Destroy*].

Application Note 12

The set of operations specified in the assignment can be collectively referred to as "access." Any subsequent use of the term "access" should be interpreted to refer to one or more of these events.

6.3.2. FDP_ACF.1 Security Attribute Based Access Control

FDP_ACF.1 Security Attribute Based Access Control

FDP_ACF.1.1

The TSF shall enforce the [*Access Control SFP*] to objects based on the following: [*subjects (defined in FDP_ACC.1.1) attempt to perform operations (defined in FDP_ACC.1.1) against objects (defined in FDP_ACC.1.1). Subject and object attributes may be used to determine whether the desired operations are permitted.*

The following are the SFP-relevant security attributes that are associated with the subjects and objects defined in FDP_ACC.1.1, as well as any restrictions on the attribute values:

- *S.DSC*
  - *DSC.ID*
- *S.Admin - none*
- *S.CApp*
  - *CApp.ID*
- *S.EPS*
  - *EPS.ID*
- *OB.P_SDO*
  - *SDO.ID*
  - *SDO.Type*
  - *SDO.AuthData*
  - *SDO.Reauth*

    - *SDO.Conf*
    - *SDO.Export*
    - *SDO.Integrity*
    - *SDO.Bind*
- *OB.T_SDO - same as OB.P_SDO*
- *OB.AuthData - none*
- *OB.Pstate - none*
- *OB.FAACntr - none*
- *OB.AntiReplay - none*
- *OB.Context- none*].

FDP_ACF.1.2

The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [

- *Any subject that has been authorized to perform any operation against any OB.P_SDO or OB.T_SDO object can continue to perform this operation if one of the following conditions is true:*
    - *The object's SDO.Reauth attribute has a value of 'none', indicating that re-authorization is not required for subsequent interactions with the SDO;*
    - *The object's SDO.Reauth attribute has a value of 'each use', indicating that re-authorization is required for each interaction with the SDO, and the subject has supplied valid authorization data to the TOE*
- *[assignment: rules automatically enforced by the TSF that always prohibit certain subject-object-operation actions]*
- *[assignment: rules automatically enforced by the TSF that always permit certain subject-object-operation actions]*
- *[assignment: rules automatically enforced by the TSF that conditionally permit certain subject-object-operation actions based on subject security attributes, object security attributes, or other conditions]*
- *[selection: [assignment: any configurable rules or parameters that can be modified to affect the behavior of the Access Control SFP], no configurable rules]]*].

FDP_ACF.1.3

The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [assignment: *rules, based on security attributes, that explicitly authorize access of subjects to objects*].

FDP_ACF.1.4

The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [*client applications can only access their own data, [assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]*].

Application Note 13

The expectation of this SFR is that the reader is given sufficient information to determine, for each object controlled by the TOE, the operations that can be performed on it based on the subject attempting to perform the operation, and whether this is conditional based on attribute values or any other circumstances.

It is expected that many of the subject-object-operation combinations will always be prohibited by the TSF, either because the target object is not externally modifiable or because the subject lacks the ability to perform the operation in question.

The ST author is not expected to create an exhaustive list of subject-object-operation combinations; it is sufficient to list those that are always permitted and those that are conditionally permitted with the expectation that all remaining combinations are prohibited.

FDP_ACF.1.3 and FDP_ACF.1.4 allow the ST author to optionally specify override conditions to resolve otherwise contradictory Access Control SFP rules. For example, the rule "S.Admin may always modify the SDO.Conf attribute of any OB.P_SDO or OB.T_SDO object" may be overridden by a statement in FDP_ACF.1.4 that identifies any particular SDO objects as non-modifiable regardless of subject authorizations.

The DSC may contain pre-installed SDOs. The DSC will enforce access control for pre-installed SDOs like any other SDO it contains or manages.

6.3.3. FDP_ETC_EXT.2 Propagation of SDOs

FDP_ETC_EXT.2 Propagation of SDOs

FDP_ETC_EXT.2.1

The TSF shall propagate only wrapped authorization data and wrapped SDOs such that only [selection: *the TOE, authorized users*] can access them.

Application Note 14

This SFR imposes security requirements on data being propagated (exported) outside the TOE. The "users" in the "authorized users" selection includes all roles (i.e. ADM-R, MFGADM-R, CApp-R).

6.3.4. FDP_FRS_EXT.1 Factory Reset

FDP_FRS_EXT.1 Factory Reset

FDP_FRS_EXT.1.1

The TSF shall permit a factory reset of the TOE upon: [**selection**: *activation by external interface, presentation of [assignment: types of authorization data required and reference to their specification], no actions or conditions*].

Application Note 15

If the DSC provides factory reset and requires an authorization to carry out the operation, then the ST author selects presentation of ... and fills in the authorization data accepted (e.g. a PIN or a cryptographic token based on some specification referenced in the assigned value). If the DSC provides factory reset external to the DSC without requiring authorization, then the ST author

selects activation by external interface. This selection is intended for use when the device containing the DSC takes responsibility for obtaining and checking the authorization for factory reset.

If any selection other than no actions or conditions is made in FDP_FRS_EXT.1.1, the selection-based SFR FDP_FRS_EXT.2 must be claimed.

6.3.5. FDP_ITC_EXT.1 Parsing of SDEs

FDP_ITC_EXT.1 Parsing of SDEs

FDP_ITC_EXT.1.1

The TSF shall support importing SDEs using [**selection**: *physically protected channels as specified in FTP_ITP_EXT.1, encrypted data buffers as specified in FTP_ITE_EXT.1, cryptographically protected data channels as specified in FTP_ITC_EXT.1*].

FDP_ITC_EXT.1.2

The TSF shall verify the integrity of the SDE using [**selection**: *message authentication code as specified in FCS_COP.1/CMAC, cryptographic hash as specified in FCS_COP.1/Hash, keyed hash as specified in FCS_COP.1/KeyedHash, key wrap encryption algorithm as specified in FCS_COP.1/KeyWrap, digital signature as specified in FCS_COP.1/SigVer, integrity verification supported by FDP_ITC_EXT.1.1*].

FDP_ITC_EXT.1.3

The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

FDP_ITC_EXT.1.4

The TSF shall bind SDEs to security attributes using [assignment: *list of ways the TSF generates security attributes and binds them to the SDEs*].

Application Note 16

The way the TSF checks the integrity of the SDE depends on the method of importation. For example, the encrypted data channel may provide data integrity as part of its service.

When a TSF parses an SDE, it should generate security attributes and create an SDO by binding the security attributes to the SDE.

6.3.6. FDP_ITC_EXT.2 Parsing of SDOs

FDP_ITC_EXT.2 Parsing of SDOs

FDP_ITC_EXT.2.1

The TSF shall support importing SDOs using [**selection**: *physically protected channels as specified in FTP_ITP_EXT.1, encrypted data buffers as specified in FTP_ITE_EXT.1, cryptographically protected data channels as specified in FTP_ITC_EXT.1*].

FDP_ITC_EXT.2.2

The TSF shall verify the integrity of the SDO using [**selection**: *message authentication code as*

specified in FCS_COP.1/CMAC, cryptographic hash as specified in FCS_COP.1/Hash, keyed hash as specified in FCS_COP.1/KeyedHash, key wrap encryption algorithm as specified in FCS_COP.1/KeyWrap, digital signature as specified in FCS_COP.1/SigVer, integrity verification supported by FDP_ITC_EXT.2.1].

FDP_ITC_EXT.2.3

The TSF shall use the security attributes associated with the imported user data.

FDP_ITC_EXT.2.4

The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC_EXT.2.5

The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

Application Note 17

The way the TSF checks the integrity of the SDO depends on the method of importation. For example, the encrypted data channel may provide data integrity as part of its service.

When a TSF parses an SDO, it should already have a set of security attributes. However, the TSF may modify these attributes, if authorized, to comply with security policies on the TOE.

6.3.7. FDP_RIP.1 Subset Residual Information Protection

FDP_RIP.1 Subset Residual Information Protection

FDP_RIP.1.1

The TSF shall ensure that any previous information content of a resource is made unavailable upon the [deallocation of the resource from] the following objects: [

- *SDOs*
- *SDEs*].

Application Note 18

When an SDE is a key then it is also subject to the key destruction requirements in FCS_CKM.6, depending on where and how it is stored. This SFR applies to authorization data that are SDEs and security attributes in SDOs.

6.3.8. FDP_SDC.2 Stored data confidentiality with dedicated method

FDP_SDC.2 Stored data confidentiality with dedicated method

FDP_SDC.2.1

The TSF shall ensure the confidentiality of ~~the~~ [the following user data [*authorization data, [assignment: SDEs identified in the confidential SDE list attribute of an SDO]*]] according to [assignment: **SDEs identified in the confidential SDE list attribute of an SDO**] while it is stored under the control of the TSF.

FDP_SDC.2.2

The TSF shall ensure the confidentiality of the user data specified in FDP_SDC.2.1 without user intervention.

Application Note 19

This SFR applies to SDOs with the confidential-SDE attribute set to require confidentiality, especially secret and private keys, Allowed Random Number Generators' state data, and vendor verification reference data. If SDEs do not require confidentiality, then its omission from this list indicates that confidentiality is not required. This SFR also applies to all authorization data appearing in the attribute list under SDO.AuthData as well as any administrator authorization data which may be stored implicitly.

6.3.9. FDP_SDI.2 Stored Data Integrity Monitoring and Action

FDP_SDI.2 Stored Data Integrity Monitoring and Action

FDP_SDI.2.1

The TSF shall monitor **SDOs and SDEs** controlled by the TSF for [*integrity errors*] on all objects, based on the following attributes: [**selection**: *[assignment: attribute associated with presence in protected storage], message authentication code as specified in FCS_COP.1/CMAC, cryptographic hash as specified in FCS_COP.1/Hash, keyed hash as specified in FCS_COP.1/KeyedHash, digital signature as specified in FCS_COP.1/SigVer, key wrap encryption algorithm as specified in FCS_COP.1/KeyWrap*].

FDP_SDI.2.2

Upon detection of a data integrity error, the TSF shall [

- *prohibit the use of the altered data*
- *send notification of the error where applicable*].

Application Note 20

This SFR deals with the mechanism that protects the integrity of the SDEs and security attributes within an SDO. This provides the binding data that ensures the prevention of unauthorized changes to the SDEs and attributes. It is not expected that a single key will be protected from corruption by multiple of these methods; however, a product may use one integrity-protection method for one type of key and a different method for other types of keys.

The cryptographic requirements for cryptographic hashes and digital signatures apply here.

No specific requirement is placed here on the nature of the integrity protection data, but the Security Target shall describe this protection measure, and shall identify the iteration of FCS_COP.1/CMAC, FCS_COP.1/Hash, FCS_COP.1/KeyedHash, FCS_COP.1/SigVer, or FCS_COP.1/KeyWrap that covers any cryptographic algorithm used.

The integrity protection data in FDP_SDI.2.1 is included in the list of attributes identified in FMT_MSA.1, and protects the value of the SDEs and of the SDO security attributes.

When an SDO is parsed, its integrity is checked when it is imported into the TOE.

6.4. Identification and Authentication

When a platform process requests the ability to create, use, modify, dispose of, etc., an SDE or SDO within the DSC, as a matter of policy, the DSC may expect or request authorization from the platform process, which may include authentication of the requester on whose behalf the platform process is acting. The DSC assumes the requester to be either a person, a process, or a device. The rules on how the requester formats the request will be outside the scope of this cPP. Upon request (or as a matter of an established protocol), the interface (on behalf of the user) presents to the DSC process those authorization values required to authorize execution of the event request. This may include one or more different types of authentication credentials. The DSC validates these items before acting upon the requested event. The validation may simply compare the authorization values to an expected value, or perform a more complex cryptographic protocol to verify the authenticity of the user. After validation, the DSC may then create and subsequently use an authorization value to represent the validation of these authorization values in anticipation of future requests.

Requirements related to the strength, quality, and performance of externally-generated authorization values supplied to the DSC, such as X.509 certificates and biometric templates, are all outside the scope of the DSC. It is expected that these will be generated according to best practices for the type of value and that this is met by the platform, where applicable. The DSC is only expected to enforce quality metrics on any authorization values it generates itself.

6.4.1. FIA_AFL_EXT.1 Authorization Failure Handling

FIA_AFL_EXT.1 Authorization Failure Handling

FIA_AFL_EXT.1.1

The TSF shall maintain [selection: *a unique counter for [**selection, choose one of**: each SDO, the following SDOs [assignment: list of SDOs]], one global counter covering [**selection, choose one of**: all SDOs, the following SDOs [assignment: list of SDOs*]], called the failed authorization attempt counters, that counts of the number of unsuccessful authorization attempts that occur related to authorizing access to these **SDOs**.

FIA_AFL_EXT.1.2

The TSF shall maintain a [selection, choose one of: *static, administrator configurable variable*] threshold of the minimal acceptable number of unsuccessful authorization attempts that occur related to authorizing access to these **SDOs**.

FIA_AFL_EXT.1.3

When the failed authorization attempt counters [selection, choose one of: *meets, surpasses*] the threshold for unsuccessful authorization attempts, the TSF shall [**selection, choose one of**:

- *prevent future authorization attempts for a static prescribed amount of time as determined using FPT_STM_EXT.1;*
- *prevent future authorization attempts for an administrator configurable amount of time as determined using FPT_STM_EXT.1;*
- *prevent all future authorization attempts indefinitely (i.e., lock), as described by*

FIA_AFL_EXT.2;

- *factory reset the TOE wiping out all non-permanent SDEs and SDOs, as described by FDP_FRS_EXT.2*

] for these **SDOs**.

FIA_AFL_EXT.1.4

The TSF shall increment the failed authorization attempt counter before it verifies the authorization.

Application Note 21

The TOE validates the authorization factors prior to determining whether user (administrator or client application) access to the SDE/SDO is permitted. In cases where validation of the authorization factors fails, the TOE blocks access to SDE/SDO. The TOE validates the authorization factors in such a way that it does not allow an attacker to circumvent the other requirements to gain knowledge about the SDE/SDO or other keying material that protects them from inadvertent exposure.

It is possible for the TOE to have different rules for the treatment of different SDOs or groups of SDOs. For example, some SDOs may trigger a factory reset in the event of excessive authorization failures while others may only temporarily block future authorization attempts. Iterations of this SFR for each distinct response can be used for each action the TSF can make (as defined by the selections in FIA_AFL_EXT.1.3) and the SDOs whose authorization failures will trigger these responses.

6.4.2. FIA_SOS.2 TSF Generation of Secrets

FIA_SOS.2 TSF Generation of Secrets

FIA_SOS.2.1

The TSF shall provide a mechanism to generate **authorization data** that meet [*the metric where for each authentication attempt, the probability shall be less than one in 1,000,000 that a random attempt will be successful*].

FIA_SOS.2.2

The TSF shall be able to enforce the use of TSF generated **authorization data** for [*assignment: **non-empty** list of TSF functions*].

Application Note 22

The TSF generates authorization data from a sufficiently large key space to ensure that users cannot employ random guessing as a statistically plausible method of authorizing actions within the TOE, both for a single event and over a session.

6.4.3. FIA_UAU.2 User Authentication before Any Action

FIA_UAU.2 User Authentication before Any Action

FIA_UAU.2.1

The TSF shall require each user **and SDO owner** to be successfully authenticated before **authorizing** any ~~other~~ TSF-mediated actions on behalf of that user **or SDO owner**.

Application Note 23

This SFR goes with FDP_ACF.1, which authorizes access to SDOs (i.e. authorizes operations with or on SDOs). The security policies in FDP_ACF.1 may require authentication of the subjects and owners of the SDOs before the TSF authorizes access to them. An authentication token is critical data bound to a user. Such data, when presented to the TOE and successfully verified by it, authenticates the user. The TOE may use the successful authentication of a user as an authorization to execute an action on its behalf, or to perform a requested operation on or with an SDO.

This requirement specifies the TSF exercise an authentication mechanism from FIA_UAU.5 by which the TOE authenticates the identity of the user requesting the operation and the owner of the SDO which is an object in the operation. Such authentication is necessary to authorize it to operate with the SDOs. A user could present a unique authentication token. The TSF may accept authentication tokens with no further conditioning. The TSF validates the authentication token prior to granting the authorization to perform the requested operation with the SDO. The SDO security attribute SDO.Reauth determines whether or not the TOE may authenticate the user and the SDO owner only once or each time each time it operates with the SDO.

The means of validation may vary based on the type of authentication token.

6.4.4. FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UAU.5 Multiple Authentication Mechanisms

FIA_UAU.5.1

The TSF shall provide [**selection**: *no mechanism, an authentication token mechanism, a cryptographic signature mechanism, [assignment: list of authentication mechanisms]*] to support user authentication.

FIA_UAU.5.2

The TSF shall authenticate any user's claimed identity according to the **following rule(s):** [**selection**: *all subject users and SDO owners shall successfully authenticate themselves using one of the mechanisms listed in FIA_UAU.5.1, the Prove service shall not accept "no mechanism" as an authentication method, [assignment: rules describing how each authentication mechanism provides authentication]*].

Application Note 24

This SFR describes the authentication mechanisms required for any user of any service as a precondition for providing authorization to execute the service. This includes the authentication of the owner of the SDOs of the service.

6.4.5. FIA_UAU.6 Re-Authenticating

FIA_UAU.6 Re-Authenticating

FIA_UAU.6.1

The TSF shall re-authenticate the user **for access to an SDO** under the conditions [*according to the policy specified in SDO.Reauth*].

Application Note 25

The allowed values for the SDO.Reauth attribute of an SDO are defined in FMT_MSA.3 and the SDO Attributes Initialization Table. The rules in FDP_ACF.1.2 and also ensure that the need for re-authorization has been checked before access to an SDO.

An SDO.Reauth value of 'none' indicates that no authentication of the subject user nor of the SDO owners is necessary. It also indicates that no reauthorization for operations using the SDO is necessary.

An SDO.Reauth value of policy indicates that there may be a more complicated set of circumstances that trigger a re-auth (re-authentication of the users and owners as well as re-authorization of the operation). This could be a policy of a time limit for which a user can use an SDO before re-authentication (e.g. 10 minutes or 24 hours).

When the TSF binds a user to access an SDO, this means that the TSF has authenticated the user and that the TSF authorized the user to have the right to exercise one or more of the following actions: generate the SDO, modify the SDO, including its security attributes, use the SDO in a TOE operation, propagate or duplicate the SDO for use by a device external to the DSC, or destroy the SDO. The user may not have exclusive rights to exercise the operations listed.

Policy as represented by the attributes in the SDO dictates whether or not a user authenticates itself in order to authorize access to the SDO.

It is possible that the attributes of some SDOs remain unchanged, and that the attributes of other SDOs may be changed by authorized users. Iterations of this SFR can be using to document the SDOs and how they apply to different policies.

6.5. Security Management

6.5.1. FMT_MOF_EXT.1 Management of Security Functions Behavior

FMT_MOF_EXT.1 Management of Security Functions Behavior

FMT_MOF_EXT.1.1

The TSF shall restrict the ability to perform the functions in FMT_SMF.1 to authenticated administrators.

6.5.2. FMT_MSA.1 Management of Security Attributes

FMT_MSA.1 Management of Security Attributes

FMT_MSA.1.1

The TSF shall enforce the [*Access Control SFP*] to restrict the ability to [modify] the security attributes **[assignment: *list of security attributes, to include attributes as specified in***

***Supported Methods for SDO Attributes*]** to [*the authorized identified roles as specified in Supported Methods for SDO Attributes*].

Table 11. Supported Methods for SDO Attributes

SDO Attribute	Modification Constraints
SDO.ID	Cannot be modified
SDO.Type	Cannot be modified
SDO.AuthData	[selection: *ADM-R, MFGADM-R, CApp-R*] roles that are authorized to modify SDO reference authorization data
SDO.Reauth	[selection: *ADM-R, MFGADM-R, CApp-R*] roles that are authorized to modify re-authorization conditions
SDO.Conf	[selection: *ADM-R, MFGADM-R, CApp-R*] roles that are authorized to modify the confidential SDE list
SDO.Export	[selection: *ADM-R, MFGADM-R, CApp-R*] roles that are authorized to modify the export flag
SDO.Integrity	Cannot be modified by users (maintained automatically by TSF)
SDO.Bind	Cannot be modified by users (maintained automatically by TSF)

Application Note 26

Supported Methods for SDO Attributes defines the required constraints on security attribute modification.

The assignments of authorized subjects in Supported Methods for SDO Attributes may be defined by the ST author in terms of roles or in terms of an action such as presentation of a valid authentication token of a particular type (in this case the ST author identifies in an Application Note the other SFRs that govern the action).

The TSF vendor may pre-install SDOs with default attributes. The ST author explains which attributes may be changed or are prohibited from changing based on the ADM-R and MFGADM-R roles (as applicable). The ST author also distinguishes between authorization values required to use pre-installed SDOs and authorization values required to change the attributes of pre-installed SDOs.

Supported Methods for SDO Attributes lists SDO ID as "cannot be modified". In some cases, a change in the attributes may cause a change in the SDO ID. In these cases, a change in the SDO ID causes the creation of a new SDO and possibly the loss of the old SDO.

Only authorized subjects can change the attributes of an SDO, and only as permitted in Supported Methods for SDO Attributes.

6.5.3. FMT_MSA.3 Static Attribute Initialization

This SFR deals with the initialization of the attributes of an SDO when it is created by parsing or provisioning. The generation process includes SDOs created by the TSF (provisioned) and those imported via FDP_ITC_EXT.2 (parsed).

The TSF is expected to give an SDO a set of security attributes at the time of its creation. This set is expected to include at least the following attributes:

- SDO identifier
- SDO type
- SDO reference authorization data (i.e. the data that is used when determining whether to grant access to an SDO, for each relevant mode of access, on the basis of an authorization token presented to the DSC)
- Re-authorization conditions (i.e. event after which re-authorization is required)
- Confidential-SDE list (each SDE in this list is held encrypted when the SDO is stored)
- Export Flag (indicating whether the SDO is allowed to be propagated)
- Integrity protection data
- Binding Data (created by the TOE to strongly link or associate the SDO with other entities such as the TOE itself or with other SDOs in a hierarchy such as a child to a parent).

The TSF provides the capability to protect the contents of an SDO (i.e. the set of its SDEs together with the SDO attributes) from unauthorized modification. The DSC shall check for such modifications before using the SDO or any of its SDEs.

FMT_MSA.3 Static Attribute Initialization

FMT_MSA.3.1

The TSF shall enforce the [*Access Control SFP*] to provide [selection, choose one of: *restrictive, permissive, [assignment: other property]*] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2

The TSF shall allow the [*authorized identified roles, according to Supported Methods for SDO Attributes Initialization*] to specify alternative initial values to override the default values when an object or information is created.

Table 12. Supported Methods for SDO Attributes Initialization

SDO Attribute	Authorized Override Role	Initialization Method	Allowed Values
SDO.ID	None	Import and generation process	[assignment: *range of allowed values*]
SDO.Type	None	Import and generation process	[assignment: *list of allowed types*]
SDO.AuthData	[selection: *ADM-R, MFGADM-R, CApp-R*]	Import process	[selection: *none, [assignment: list of types of authentication tokens allowed], [assignment: range of authorization values allowed]*]
	None	Generation process	

SDO Attribute	Authorized Override Role	Initialization Method	Allowed Values
SDO.Reauth	None	Import and generation process	[selection: *none, each access, policy*]
SDO.Conf	None	Import and generation process	[assignment: *list of SDEs of which the TOE must provide a confidentiality service*]
SDO.Export	None	Import and generation process	[selection: *exportable, non-exportable*]
SDO.Integrity	None	Import and generation process	[assignment: *range of allowed values*]
SDO.Bind	None	Import and generation process	[assignment: *range of allowed values*]

Application Note 27

The Supported Methods for SDO Attributes Initialization Table is referenced from FMT_MSA.3 and matches the attributes covered by FMT_MSA.1 (which defines controls on the modification of the attributes). The initialization of these security attributes occurs when an SDO is either parsed by the TOE or generated on the TOE.

An imported object contains the default values for each attribute, where allowed. The TSF can override default values for the following attributes of imported objects: SDO.ID, SDO.Type, SDO.Reauth, SDO.Export, and SDO.Integrity. The TSF may override default values in these cases to force the objects to comport to established structures within the TOE, or to comply with TOE-wide security policies. In these cases, the defined roles cannot override the default values. For SDO.AuthData, the TSF shall allow the CApp-R role to override authorization data that may arrive with the object. For SDO.Conf the TSF accepts the imported value for this attribute. SDO.Bind is explained below.

Unless otherwise noted, both the ADM-R and CApp-R roles can initiate the generation process. The ADM-R and CApp-R roles will provide the default values for the attributes. The TSF checks SDO.Type, SDO.AuthData, SDO.Reauth, SDO.Conf, and SDO.Export for compliance with established security policies and refuses to create objects which do not comply with overriding the value of any of these attributes. In the cases of the SDO.ID and SDO.Integrity, the TSF generates these values and therefore there is no need or ability to override.

In the case of SDO.Bind for both import and generation processes, the TSF may override values that denote a binding to the TOE, but not those values that denote a binding to other keys. In the case of the import process, the defined roles cannot override the default values for any binding.

The SDO.AuthData attribute is data that is required in order to validate authorization of a subject to access the SDO (in each of the modes relevant to that SDO). The nature of this data will depend on the authorization mechanism used in the TOE, as described in FIA_UAU.2.

The SDO.Reauth attribute for an individual SDO specifies the conditions where access to the SDO will require reauthorization to continue access to the SDO. Examples of TOE-specified events might be explicit revocation of authorization by a user, expiry of a time interval, or completion of a fixed

number of uses since the last authorization. The re-authorization conditions are used in FIA_UAU.6 and FDP_ACF.1. These determine whether a single authorization by the SDO owner will allow any number of uses of the SDO until the end of the user's session (value 'none'), or whether each use of the SDO must be individually authorized (value 'each access'), or whether re-authorization must happen each time one of the TOE-specified events occurs.

The SDO.Conf attribute indicates which SDEs, if any, the TOE encrypts when not in operational use. The TOE should use the methods in FCS_COP.1/AEAD, FCS_COP.1/SKC, FCS_STG_EXT.1, or FCS_CKM_EXT.3 to protect the SDEs in this list.

The SDO.Integrity attribute includes evidence that the TSF can use to protect and verify the integrity of the SDO.

Attributes assigned by the TOE to any parsed SDOs are described in the ST. Where attributes may be assigned by authorized roles, the information should also be described in operational user guidance.

The TOE uses the Binding Data for an SDO to strongly link the SDO to the TOE, a parent SDO in a hierarchy, or to nothing at all. SDOs bound to nothing may freely travel from one TOE to another without restrictions. If bound to another SDO as a child to a parent in a hierarchy, it may travel only where the parent SDO travels. If bound to the TOE, it may travel to any other TOE for any reason, even if the TOE moves its parent to another TOE. Note that vendors will initialize attributes of pre-installed SDOs with default values. However, authorization values to change the attributes of pre-installed SDOs may differ from the authorization value required to use the pre-installed SDO.

In cases in which the SDO.ID is a cryptographic hash of the attributes and SDEs, that value may act as both SDO.ID and SDO.Integrity for the SDO.

When a remote peer sends an SDO to the TOE, it properly indicates through the SDE-confidentiality list of any authorization values and authentication tokens present in the SDO, whether they are present in the SDE or as attributes, which control access to the SDE.

When a TOE generates an SDO internally for the first time, it properly indicates through the SDE-confidentiality list any SDEs that are authorization values or authentication tokens. Similarly, if any of the attributes are authorization values or authentication tokens, the TOE will properly indicate through the SDE-confidentiality list that it will encrypt them prior to storing them.

6.5.4. FMT_SMF.1 Specification of Management Functions

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1

The TSF shall be capable of performing the following management functions: [

- *Reset TOE to factory state for FDP_FRS_EXT.1*
- *Configure authorization policies for TOE resources*

  [selection:

  - *set authorization failure parameters for FIA_AFL_EXT.1,*

- *update TOE firmware,*
- *update pre-installed SDOs,*
- *unlock access to SDO following excessive failed authorization attempts,*
- *no other functions]*].

Application Note 28

If FPT_MFW_EXT.1 selects mutable firmware, then FMT_SMF.1 must select update TOE firmware and update pre-installed SDOs.

Recall that resetting a TOE to factory state also wipes all user data, but may not wipe out pre-installed SDOs. Configuring authorization policies includes setting policies for allowed access to SDOs.

Protections for pre-installed SDEs/SDOs come through the firmware as well as through firmware updates. In the same vein, the authorized updates may also affect the SDEs as well, if the vendor so chooses. One could say that the authorized update binds the attributes present in the functionality of the firmware to the pre-installed SDEs.

6.5.5. FMT_SMR.1 Security Roles

FMT_SMR.1 Security Roles

FMT_SMR.1.1

The TSF shall maintain the roles: [*ADM-R, MFGADM-R, CApp-R*].

FMT_SMR.1.2

The TSF shall be able to associate users with roles.

Application Note 29

This cPP uses the term "user" throughout to reference the ADM-R, MFGADM-R and CApp-R roles simultaneously.

6.6. Protection of the TSF

6.6.1. FPT_FLS.1/FI Failure with Preservation of Secure State (Fault Injection)

FPT_FLS.1/FI Failure with Preservation of Secure State (Fault Injection)

FPT_FLS.1.1/FI

The TSF shall preserve a secure state when the following types of failures occur: [*fault injections*].

Application Note 30

Note that a secure state does not imply the uninterrupted enforcement of all claimed security functionality it is appropriate for the TSF to "fail closed" and block the execution of security-relevant behavior if a fault injection attempt or other significant glitch occurs.

6.6.2. FPT_MFW_EXT.1 Mutable/Immutable Firmware

FPT_MFW_EXT.1 Mutable/Immutable Firmware

FPT_MFW_EXT.1.1

The TSF shall be maintained as [selection:

- *immutable firmware*
- *mutable firmware conforming to FPT_FLS.1/FW, FPT_MFW_EXT.2, FPT_MFW_EXT.3, and FPT_RPL.1/Rollback*].

6.6.3. FPT_MOD_EXT.1 Debug Modes

FPT_MOD_EXT.1 Debug Modes

FPT_MOD_EXT.1.1

The TSF shall provide no access to debug modes.

Application Note 31

'Debug modes' may include, but are not limited to, any alternate mode of operation, such as developer mode, test mode, manufacturer mode, or altered boot mode. These modes may be available in some versions of the TOE, but not in the final production version.

6.6.4. FPT_PHP.3 Resistance to Physical Attack

FPT_PHP.3 Resistance to Physical Attack

FPT_PHP.3.1

The TSF shall resist [*data extraction via fault injection from extreme temperatures and abnormal voltage*] to the [*TSF storage elements that contain [selection: SDEs, SDOs, firmware]*] by responding automatically such that the SFRs are always enforced.

Application Note 32

Physical protection mechanisms as envisioned by this requirement are mechanisms that protect communications to the extent that encryption or other logical protections are not required to ensure confidentiality, integrity, and assured identification of endpoints. Such mechanisms may include, for example, physically isolated traces, or mechanisms that take advantage of physical properties of signals to ensure that communications are receivable only by the intended endpoint.

Any physical external casing or potting material of the TOE is considered an 'external interface', not just those interfaces over which data is transmitted. This ensures that the TSF will respond appropriately if, for example, an attacker penetrates the physical surface of the DSC in an attempt to access its stored data.

The TOE's protection against abnormal temperature and voltage can be considered equivalent to what is required by assertion AS07.77 of [ISO-TR].

6.6.5. FPT_PRO_EXT.1 Root of Trust

FPT_PRO_EXT.1 Root of Trust

FPT_PRO_EXT.1.1

The TSF shall contain an SDO that contains the identity of the Root of Trust.

Application Note 33

Every DSC is expected to have a single RoT that comprises the DSC hardware and pre-installed SDOs, from which services (e.g. Storage, Authorization, etc.) can be offered.

Depending on the use case and the way status registers are used, unique identity keys may be bound to the TOE, the TOE platform, or both.

The sole presence of unique identity keys linking to the RoT does not prove authenticity without the use of digital signatures.

FPT_PRO_EXT.1.2

The TSF shall maintain Root of Trust data as [selection: *immutable, mutable if and only if its mutability is controlled by a unique identifiable owner*].

Application Note 34

One expects that only authorized sources can modify the single RoT, such as through a secure update.

The process of authenticating the source of a secure update may involve querying the identity of the manufacturer, contained on a pre-installed SDO. If this identity is in the form of an X.509 certificate containing a signature verification key signed by the manufacturer, then the authentication process is sufficient.

A unique identifiable owner is assumed to be one with an ADM-R role; however, there may be circumstances where the owner does not take on an ADM-R role, which should be documented.

6.6.6. FPT_ROT_EXT.1 Root of Trust Services

FPT_ROT_EXT.1 Root of Trust Services

FPT_ROT_EXT.1.1

The TSF shall provide a Root of Trust for Storage, a Root of Trust for Authorization, and [selection: *Root of Trust for Measurement, Root of Trust for Reporting, no others*] based on the Root of Trust identified in FPT_PRO_EXT.1.1.

Application Note 35

This document uses the [GP_ROT] definitions for RoT for Storage (denoted as the combination of RoT for Confidentiality and RoT for Integrity), Authorization, Measurement, and Reporting. DSCs use Roots of Trust for Storage to protect SDOs. Section 6.4 has a number of requirements for ensuring the TSF has functionality to authorize a user in order to access an SDO, including FIA_UAU.6.

If both Root of Trust for Measurement and Root of Trust for Reporting are selected in FPT_ROT_EXT.1.1, the selection-based SFR FDP_DAU.1/Prove must be claimed.

6.6.7. FPT_ROT_EXT.2 Root of Trust for Storage

FPT_ROT_EXT.2 Root of Trust for Storage

FPT_ROT_EXT.2.1

The TSF shall prevent unauthorized access to SDOs associated with the Root of Trust for Storage.

Application Note 36

TOEs may use shielded locations or cryptographic protections to prevent unauthorized access to SDOs. Unauthorized access includes unauthorized disclosure of secret SDOs (e.g. secret keys, private keys) and unauthorized modification of both secret and non-secret SDOs (e.g. public keys, certificates). Use FDP_SDC.2 to protect the confidentiality of secret SDOs associated with the RoT for Storage. Use FDP_SDI.2 to protect the integrity of SDOs associated with the RoT for Storage.

6.6.8. FPT_RPL.1/Authorization Replay Prevention

FPT_RPL.1/Authorization Replay Prevention

FPT_RPL.1.1/Authorization

The TSF shall detect replay for the following entities: [*user authorization of operations on SDOs*].

FPT_RPL.1.2/Authorization

The TSF shall perform [*denial of the requested operation on the SDO*] when a replay is detected **using the following methods [selection: *monotonic counters, random nonces, [assignment: other methods as specified]*]**.

Application Note 37

The TSF receives authorization from an external source to the TOE to perform an operation on an SDO. If the operation on the SDO is restricted to authorized users, then anyone observing the communication to the TOE can copy the authorization and replay it. Random nonces and monotonic counters are but two mechanisms the TSF can use to mitigate replay. In this requirement, operations on SDOs include generating, using, modifying, propagating, and destroying. Besides monotonic counters and random nonces, the TSF could employ other methods to prevent replay of user authorizations, which the Security Target should describe.

6.6.9. FPT_TST.1 TSF Testing

FPT_TST.1 TSF Testing

FPT_TST.1.1

The TSF shall run a suite of the following self-tests **during power-on start-up** [selection: periodically during normal operation, at the request of the authorized user, **at no other condition,** at the conditions [*assignment: conditions under which self-test should occur*]] to demonstrate the correct operation of [the TSF]: [assignment: *list of self-tests run by the TSF*].

FPT_TST.1.2

The TSF shall provide authorized users with the capability to verify the integrity of [TSF data].

FPT_TST.1.3

The TSF shall provide authorized users with the capability to verify the integrity of **the** [TSF].

Application Note 38

This requirement intends to cover integrity of the TSF functionality (i.e. runtime checks).

TSF integrity testing provides the ability to test the TSF's correct operation. These tests are expected to be performed automatically and autonomously at start-up but may also be performed periodically during operation, at the request of the authorized user, or when other conditions are met. It also provides the ability to verify the integrity of TSF data and executable code.

All cryptographic functions come with known answer tests (KATs). In addition to verifying the integrity of the firmware executing the TSF, the DSC should also verify the integrity of any data associated with the TSF (such as constants for cryptographic algorithms) as well as performing the KATs.

6.7. Resource Utilization

6.7.1. FRU_FLT.1 Degraded Fault Tolerance

FRU_FLT.1 Degraded Fault Tolerance

FRU_FLT.1.1

The TSF shall ensure the operation of [*protection of TSF data*] when the following failures occur: [*fault injection*].

Application Note 39

TSF data may be protected in response to a fault injection either by providing a method to ensure that the data remains protected or by logically destroying the data or any part of a key chain that encrypts it. This behavior may differ based on the type of fault.

6.8. TOE Security Functional Requirements Rationale

The following rationale provides justification for each security problem definition (SPD) aspect of the TOE, showing that the SFRs are suitable to meet and achieve the SPD aspect - this mapping follows CC:2022 Part 1 Appendix B.5:

Table 13. SFR Rationale

SPD	Addressed by	Rationale
T.BRUTE_FORCE_AUTH	FIA_AFL_EXT.1	This requirement enforces authentication failure handling capabilities to ensure that brute force attacks on the TSF are not possible.
	FIA_SOS.2	This requirement protects against brute force authentication by generating secrets that are statistically difficult to guess.
	FPT_STM_EXT.1	This requirement provides reliable system time services that may be used to determine when excessive authentication failure attempts have been made.
	FIA_AFL_EXT.2 (selection-based)	This requirement defines how access to an SDO is restored if excessive authentication failures trigger a lock on it.

SPD	Addressed by	Rationale
T.UNAUTHORIZED_ACCESS	FCS_STG_EXT.1	This requirement ensures that key data is placed into protected storage and cannot be modified by untrusted subjects.
	FDP_ACC.1	This requirement defines an access control policy that governs the authorization required to interact with SDOs.
	FDP_ACF.1	This requirement defines the rules enforced by the access control policy defined in FDP_ACC.1 to control access to SDOs.
	FDP_ETC_EXT.2	This requirement ensures that protected data propagated outside the TOE is not disclosed to any unauthorized subjects.
	FIA_UAU.2	This requirement defines the methods by which users authenticate to the TOE to prove their identity prior to interacting with any protected data.
	FIA_UAU.5	This requirement provides the TSF with the ability to specify the use of multiple authentication mechanisms as a prerequisite to granting access to protected functions or data.
	FIA_UAU.6	This requirement defines when authorization checks are performed for user requests to access SDOs.
	FMT_MOF_EXT.1	This requirement enforces access control on the management functions provided by the TOE.
	FMT_MSA.1	This requirement enforces restrictions on the subjects that can interact with SDOs and their attributes.
	FMT_MSA.3	This requirement defines the default access restrictions that are enforced on SDO attributes if not overridden by specific access control policy rules.

SPD	Addressed by	Rationale
T.UNAUTHORIZED_ACCESS	FMT_SMF.1	This requirement defines the management functions that are provided by the TOE to authorized subjects.
	FMT_SMR.1	This requirement defines the roles used by the TSF for enforcement of access control to protected functions and data.
	FPT_MOD_EXT.1	This requirement ensures that there are no accessible debug modes that could be used to circumvent access control policy restrictions preventing a user from accessing protected functions or data.
	FPT_PHP.3	This requirement ensures that some mechanism is in place to thwart unauthorized attempts to access protected functions or data through physical tampering of the TOE.
	FPT_PRO_EXT.1	This requirement defines the RoT for the TOE, which is used to derive all access control functionality.
	FPT_ROT_EXT.2	This requirement enforces the RoT for Storage to enforce access control against SDOs.
	FPT_RPL.1/Authorization	This requirement ensures that access control restrictions cannot be bypassed through replay of operations.
	FIA_AFL_EXT.2 (selection-based)	This requirement defines the access control that is enforced on an SDO if excessive authentication failures block access to it.

SPD	Addressed by	Rationale
T.HW_ATTACK	FDP_RIP.1	This requirement ensures that any purged SDEs/SDOs are erased in residual memory so that their future recovery is prevented.
	FPT_FLS.1/FI	This requirement ensures that fault injections cannot be used to circumvent access control policy restrictions preventing a user from accessing protected functions or data.
	FPT_PHP.3	This requirement ensures that some mechanism is in place to thwart physical tampering of the TOE.
	FRU_FLT.1	This requirement ensures that fault injection attempts do not interfere with the enforcement of access control against protected data.
	FDP_FRS_EXT.2 (selection-based)	This requirement ensures that all user-specific SDOs are purged upon factory reset and may indicate any factory default SDOs that are reset to their initial values.

SPD	Addressed by	Rationale
T.SDE_TRANSIT_COMPROMISE	FCS_CKM.6	This requirement ensures that key data is destroyed in a manner that prevents its future recovery.
	FCS_COP.1/AEAD (selection-based)	This requirement provides a cryptographic operation for maintaining the confidentiality and integrity of SDOs.
	FCS_COP.1/SKC	This requirement provides a cryptographic operation for maintaining the confidentiality of SDOs.
	FDP_ETC_EXT.2	This requirement ensures that the confidentiality of protected data propagated outside the TOE is maintained.
	FDP_FRS_EXT.1	This requirement defines the condition in which a factory reset will be initiated, which triggers a purge of stored SDEs.
	FDP_ITC_EXT.1	This requirement ensures that all SDEs parsed by the TOE are transmitted over a secure channel.
	FDP_ITC_EXT.2	This requirement ensures that all SDOs parsed by the TOE are transmitted over a secure channel.
	FDP_SDC.2	This requirement ensures that the confidentiality of authorization data is protected prior to storage.
	FPT_ITT.1 (optional)	This requirement ensures that confidentiality and integrity is maintained in cases where data is transmitted between physically separate parts of a distributed TOE.
	FTP_ITC_EXT.1 (selection-based)	This requirement defines a cryptographically protected channel that the TSF can use to securely parse data being imported into it.
	FTP_ITE_EXT.1 (selection-based)	This requirement defines the cryptographic method used to transfer data between the TOE and external entities.
	FTP_ITP_EXT.1 (selection-based)	This requirement defines a physically protected channel that the TSF can use to securely parse data being imported into it.

SPD	Addressed by	Rationale
T.WEAK_ELEMENT_BINDING	FCS_COP.1/Hash	This requirement provides a cryptographic operation for asserting the integrity of SDOs.
	FCS_COP.1/KeyedHash	This requirement provides a cryptographic operation for asserting the authenticity of SDOs.
	FCS_COP.1/SigGen	This requirement provides a cryptographic operation for preserving the authenticity of SDOs.
	FCS_COP.1/SigVer	This requirement provides a cryptographic operation for asserting the authenticity of SDOs.
	FDP_SDI.2	This requirement ensures that SDEs/SDOs are monitored for integrity violations.
	FPT_TST.1	This requirement defines the mechanisms used to verify and attest to the integrity of the TSF.
	FPT_PRO_EXT.2 (optional)	This requirement ensures that the TSF can measure the integrity of its stored data.
	FCS_COP.1/CMAC (selection-based)	This requirement provides a cryptographic operation for asserting the authenticity of SDOs.
	FPT_FLS.1/FW (selection-based)	This requirement requires the TSF to take action to preserve its secure operation if any violations to its firmware integrity are detected.
T.WEAK_OWNERSHIP_BINDING	FDP_ITC_EXT.1	This requirement ensures that all SDEs parsed by the TOE have verifiable integrity.
	FDP_ITC_EXT.2	This requirement ensures that all SDOs parsed by the TOE have verifiable integrity.
	FDP_SDC.2	This requirement ensures that SDEs/SDOs are stored with confidentiality and that all authorization data is protected prior to storage.
	FPT_ROT_EXT.1	This requirement defines the RoT services that are available for the protection of data.
	FDP_ITC_EXT.1	This requirement ensures that all SDEs parsed by the TOE include appropriate binding metadata.
	FPT_ROT_EXT.3 (optional)	This requirement allows the TSF to provide a RoT for Reporting that can provide assured information about the stored SDEs.
	FDP_DAU.1/Prove (selection-based)	This requirement defines the Prove service that can be used to invoke the Roots of Trust for Measurement and Reporting and provide affirmation of the validity of TSF and stored data.

SPD	Addressed by	Rationale
T.UNAUTH_UPDATE	FPT_MFW_EXT.1	This requirement specifies whether the TOE's firmware is mutable or immutable.
	FPT_FLS.1/FW (selection-based)	This requirement requires the TSF to take action to preserve its secure operation if a rollback attempt or invalid firmware update is detected.
	FPT_MFW_EXT.2 (selection-based)	This requirement ensures that the TSF can generate evidence that its mutable firmware integrity remains intact.
	FPT_MFW_EXT.3 (selection-based)	This requirement ensures that any firmware updates to the TSF are genuine.
	FPT_RPL.1/Rollback (selection-based)	This requirement ensures that the TSF will not permit rollback attempts of its firmware.
T.WEAK_CRYPTO	FCS_CKM.1	This requirement specifies the supported methods of key generation.
	FCS_CKM.2	This requirement specifies the supported methods of key distribution.
	FCS_CKM_EXT.7	This requirement ensures the use of strong key agreement mechanisms.
	FCS_COP.1/AEAD (selection-based)	This requirement ensures the use of strong methods to encrypt and authenticate sensitive data.
	FCS_COP.1/Hash	This requirement ensures the use of strong hash mechanisms.
	FCS_COP.1/KeyedHash	This requirement ensures the use of strong HMAC mechanisms.
	FCS_COP.1/SigGen	This requirement ensures the use of strong digital signature services.
	FCS_COP.1/SigVer	This requirement ensures the use of strong digital signature services.
	FCS_COP.1/SKC	This requirement ensures the use of strong methods to encrypt sensitive data.
	FCS_RBG.1	This requirement ensures the use of strong random bit generation mechanisms.
	FCS_OTV_EXT.1	This requirement ensures that one-time values used by the TOE do not negatively impact key strength.

SPD	Addressed by	Rationale
T.WEAK_CRYPTO	FPT_STM_EXT.1	This requirement provides reliable system time services that may be used as inputs to cryptographic functions.
	FCS_RBG.2 (optional)	This requirement provides an external interface to seed the random bit generator that enforces strong cryptography by requiring a minimum amount of input.
	FCS_RBG.3 (optional)	This requirement provides an internal interface to seed the random bit generator that enforces strong cryptography by requiring a minimum amount of input.
	FCS_RBG.4 (optional)	This requirement provides multiple internal interfaces to seed the random bit generator that enforces strong cryptography by requiring a minimum amount of input.
	FCS_RBG.5 (optional)	This requirement ensures that combining multiple sources of entropy are combined in a way that enforces strong cryptography by requiring a minimum amount of input to the random bit generator.
	FCS_RBG.6 (optional)	This requirement provides an interface to access RBG output so that the TSF can support the use of strong cryptography in its operational environment.
	FCS_CKM.1/AKG (selection-based)	This requirement ensures the generation of strong asymmetric keys.
	FCS_CKM.1/SKG (selection-based)	This requirement ensures the generation of strong symmetric keys.
	FCS_CKM_EXT.3 (selection-based)	This requirement ensures the use of strong methods to perform key encapsulation, key wrawpping, or key encryption when accessing keys.
	FCS_CKM.5 (selection-based)	This requirement ensures the use of strong mechanisms to perform key derivation.

SPD	Addressed by	Rationale
T.WEAK_CRYPTO	FCS_COP.1/CMAC (selection-based)	This requirement ensures the use of strong methods to perform CMAC.
	FCS_COP.1/KeyEncap (selection-based)	This requirement ensures the use of strong methods to perform key encapsulation.
	FCS_COP.1/KeyWrap (selection-based)	This requirement ensures the use of strong methods to perform key wrapping.
	FCS_COP.1/XOF (selection-based)	This requirement ensures the use of strong methods to perform XOF.
	FCS_CKM_EXT.8 (selection-based)	This requirement ensures the use of strong methods to derive keys from password data.

Chapter 7. Security Assurance Requirements

This cPP identifies the Security Assurance Requirements (SARs) to frame the extent to which the evaluator assesses the documentation applicable for the evaluation and performs independent testing.

This section lists the set of SARs from CC part 3 that are required in evaluations against this cPP. Individual Evaluation Activities to be performed are specified in [DSC SD].

The general model for evaluation of TOEs against STs written to conform to this cPP is as follows:

After the ST has been approved for evaluation, the ITSEF will obtain the TOE, supporting environmental IT (if required), and the administrative/user guides for the TOE. The ITSEF is expected to perform actions mandated by the Common Evaluation Methodology (CEM) and the Evaluation Activities contained within the SD.

The actions for ALC Class (ALC_CMC.1 and ALC_CMS.1) are specified solely within the CEM, while the remaining Assurance Classes are extended beyond the CEM as described in the SD. The SD is intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The Evaluation Activities that are captured in the SD also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the cPP.

Table 14. Security Assurance Requirements

Assurance Class	**Assurance Components**
Security Target (ASE)	Conformance Claims (ASE_CCL.1)
	Extended Components Definition (ASE_ECD.1)
	ST Introduction (ASE_INT.1)
	Security Objectives for the Operational Environment (ASE_OBJ.1)
	Direct Rationale Security Requirements (ASE_REQ.1)
	Security Problem Definition (ASE_SPD.1)
	TOE Summary Specification (ASE_TSS.1)
Development (ADV)	Basic Functional Specification (ADV_FSP.1)
Guidance Documents (AGD)	Operational User Guidance (AGD_OPE.1)
	Preparative Procedures (AGD_PRE.1)
Life-cycle Support (ALC)	Labelling of the TOE (ALC_CMC.1)
	TOE CM Coverage (ALC_CMS.1)
Tests (ATE)	Independent Testing - Conformance (ATE_IND.1)
Vulnerability Assessment (AVA)	Vulnerability Survey (AVA_VAN.1)

7.1. ASE: Security Target

The ST is evaluated as per ASE activities defined in the CEM. In addition, there may be Evaluation Activities specified within the SD that call for necessary descriptions to be included in the TSS that are specific to the TOE technology type.

In addition to using the ST to demonstrate that ASE_TSS.1 has been satisfied, this cPP requires the creation of supplemental documentation to justify how the TOE satisfies certain SFRs. This documentation is separated from the ST because the required level of detail may include information that is proprietary to the developer of the TOE. The required supplemental documentation includes entropy documentation and key management documentation. The requirements for the entropy documentation are described in Appendix D, *Entropy Documentation and Assessment* of this cPP. The requirements for the key management documentation are described in the SD under the SFRs that require a detailed description of the TSF's key management.

7.2. ADV: Development

The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any additional information required by this cPP that is not to be made public (e.g., Entropy Essay). The DSC cPP requires only basic functional specification of interfaces presented in the AGD documentation (see Section 7.2.1) and specification of interfaces that can be invoked by a dependent component in a composed evaluation where the DSC is the base component.

7.2.1. Basic Functional Specification (ADV_FSP.1)

The functional specification describes the TOE Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces. For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation.

The Evaluation Activities in the SD are associated with the applicable SFRs; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

7.2.2. Specification of DSC Interface for Use in Composite Evaluations

For the DSC to serve as a base component in a composed evaluation, all DSC interfaces that may be invoked by a dependent component to satisfy dependent component SFRs must be documented.

A DSC that complies with this cPP must make services available to a dependent component through interfaces. The DSC ST author must describe these interfaces in order for a dependent component evaluation to properly map the DSC-provided services to SFRs within the dependent component PP, and to ensure that dependent component implementations properly use the service interfaces.

The Evaluation Activities in the SD require specifying each such interface exported.

7.3. AGD: Guidance Documentation

The guidance documents will be provided with the ST. Guidance must include a description of how the IT personnel verifies that the Operational Environment can fulfill its role for the security functionality. The documentation should be in an informal style and readable by the IT personnel.

Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes:

- instructions to successfully install the TSF in that environment; and
- instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and
- Instructions to provide a protected administrative capability.

Guidance pertaining to particular security functionality must also be provided; requirements on such guidance are contained in the Evaluation Activities specified in the SD.

7.3.1. Operational User Guidance (AGD_OPE.1)

The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages.

The developer should review the Evaluation Activities contained in the SD to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

7.3.2. Preparative Procedures (AGD_PRE.1)

As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

7.4. Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this cPP, life-cycle support is limited to end-user-visible aspects of the life cycle, rather than an examination of the TOE vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it is a reflection on the information to be made available for evaluation at this assurance level.

7.4.1. Labelling of the TOE (ALC_CMC.1)

This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. The evaluator performs the CEM work units associated with ALC_CMC.1

7.4.2. TOE CM Coverage (ALC_CMS.1)

Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator

performs the CEM work units associated with ALC_CMS.1.

7.5. Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. For this cPP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirement.

7.5.1. Independent Testing - Conformance (ATE_IND.1)

Testing is performed to confirm the functionality described in the TSS as well as the operational guidance (includes "evaluated configuration" instructions). The focus of the testing is to confirm that the requirements specified in Section 6 are being met. The Evaluation Activities in the SD identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

7.6. Class AVA: Vulnerability Assessment

For the current generation of this cPP, the iTC is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products and provide that content into the AVA_VAN discussion. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. This information will be used in the development of future Protection Profiles.

7.6.1. Vulnerability Survey (AVA_VAN.1)

As with ATE_IND, the evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to determine the vulnerabilities that have been found in components similar to the component under evaluation (such as a secure element) and when applicable, implemented communication protocols in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Appendix A: Optional Requirements

A.1. Cryptographic Support

A.1.1. FCS_RBG.2 Random Bit Generation (External Seeding)

FCS_RBG.2 Random Bit Generation (External Seeding)

FCS_RBG.2.1

The TSF shall be able to accept a minimum input of [assignment: *minimum input length greater than zero*] from a TSF interface for the purpose of **obtaining entropy**.

Application Note 40

In order to maintain compliance with NIST SP 800-90A Revision 1, the TSF accepts enough bits input from an external entropy source to satisfy the entropy requirements of the DRBG.

The TSF interface for the purpose of seeding here is the interface used to gather entropy for initializing the seed.

A.1.2. FCS_RBG.3 Random Bit Generation (Internal Seeding - Single Source)

FCS_RBG.3 Random Bit Generation (Internal Seeding - Single Source)

FCS_RBG.3.1

The TSF shall be able to seed the **DRBG** using a [selection: choose one of: *TSF software-based **entropy** source, TSF hardware-based **entropy** source*] [assignment: *name of **entropy** source*] with [assignment: *number of bits*] bits of min-entropy.

Application Note 41

If an ST Author wishes to use multiple internal entropy sources, they iterate this requirement for each entropy source used by the TSF.

A.1.3. FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)

FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)

FCS_RBG.4.1

The TSF shall be able to seed the **DRBG** using [selection: *[assignment: number] TSF software-based **entropy** source(s), [assignment: number] TSF hardware-based **entropy** source(s)*].

A.1.4. FCS_RBG.5 Random Bit Generation (Combining Entropy Sources)

FCS_RBG.5 Random Bit Generation (Combining Entropy Sources)

FCS_RBG.5.1

The TSF shall [**selection**: *hash, concatenate and hash, xor, input into a linear feedback shift*

register, [assignment: combining operation]] [selection: *output from TSF **entropy** source(s), input from TSF interface(s) for **obtaining entropy***] to create the entropy input into the derivation function as defined in [**selection**: *ISO/IEC 18031: 2011, NIST SP 800-90A Revision 1*] resulting in a minimum of [assignment: *number of bits*] bits of min-entropy.

Application Note 42

One can apply NIST SP 800-90B (or AIS-31) statistical tests against internal entropy sources (a.k.a. raw entropy) to confirm the min-entropy of the entropy sources either in aggregate or individually. NIST SP 800-90B (or AIS-31) statistical tests against external entropy sources are not needed since the TOE is unable to enforce entropy requirements or conditioning requirements against external sources of entropy. However, the TSS may include estimates for min-entropy from external sources that contribute to the overall entropy requirements for either the DRBG or for FCS_OTV_EXT.1.

FCS_RBG.5 specifies the combining operation such that the combined min-entropy of all the internal sources and the estimated entropy of the external sources is greater than or equal to the desired entropy of the output of the combining operation. The output could be used as a nonce, or a seed for a DRBG.

The TSF interface(s) for seeding here is the interface used to gather entropy for initializing the seed.

A.1.5. FCS_RBG.6 Random Bit Generation Service

FCS_RBG.6 Random Bit Generation Service

FCS_RBG.6.1

The TSF shall provide a [selection: *hardware, software, [assignment: other interface type]*] interface to make the **DRBG** output, as specified in FCS_RBG.1 Random Bit Generation (RBG), available as a service to entities outside of the TOE.

A.2. Protection of the TSF

A.2.1. FPT_ITT.1 Basic Internal TSF Data Transfer Protection

FPT_ITT.1 Basic Internal TSF Data Transfer Protection

FPT_ITT.1.1

The TSF shall protect TSF data from [disclosure] **and [selection: *modification, no other actions*]** when it is transmitted between separate parts of the TOE.

A.2.2. FPT_PRO_EXT.2 Data Integrity Measurements

FPT_PRO_EXT.2 Data Integrity Measurements

FPT_PRO_EXT.2.1

The TSF shall be able to quantify the integrity of the data protected by the TOE by generating integrity measurements and assertions.

Application Note 43

The generation of these integrity measurements and assertions is the creation of OB.Pstate.

Data protected by the TOE includes DSC firmware, DSC configuration data, and user data. DSC configuration data may include permanent SDEs or SDOs such as immutable or mutable root keys, authorization values, and authentication tokens (i.e. DSC.ID, OB.P_SDO, OB.FAACntr, OB.AntiReplay, and OB.Context). User data may include transient SDEs and SDOs as well as authorization values and authentication tokens bound to these SDEs and SDOs (i.e. OB.T_SDO).

FPT_PRO_EXT.2.2

The TSF shall accumulate platform characteristics using a consistent [assignment: *description of process for accumulating platform characteristics*] process in which verified quantifiable measurements and assertions are accumulated by the RoT for Measurement to prove the integrity of its SDOs.

Application Note 44

Although a platform may enter any state possible — including undesirable or insecure states — it can use platform characteristics, including integrity measurements and assertions, along with logging and reporting to accurately report the state derived from data attributing to those states. In this context, platform characteristics can include, but is not limited to, cryptographic hashes of binary data, security-critical configurations, register values (including status registers) and milestones, such as verification of firmware, or transitioning from a boot phase to an operational phase. A platform characteristic may also represent the state of some entity outside the DSC. A process independent from the DSC or the host containing the DSC may evaluate the platform characteristics and determine an appropriate action.

A.2.3. FPT_ROT_EXT.3 Root of Trust for Reporting Mechanisms

FPT_ROT_EXT.3 Root of Trust for Reporting Mechanisms

FPT_ROT_EXT.3.1

The TSF shall be able to attest to a state as represented by platform characteristics with a Root of Trust for Reporting mechanism that uses for its identity [selection: *a cryptographically verifiable identity in FPT_PRO_EXT.1, an alias key bound to the cryptographically verifiable identity in FPT_PRO_EXT.1*] and using a signature algorithm as specified in FCS_COP.1/**SigGen**.

Application Note 45

While it is possible for a group of components to share a single unique group identifier, it is important to ensure that individual components have their own unique identifiers relative to each other.

Resident keys or aliases are designed such that they are never visible outside the subset of DSC scope containing the RoT services and are only to be used for encryption. Therefore, possession of such aliases or keys can only be proved indirectly by using it to decrypt a value that has been encrypted with a corresponding public key. In this way, these resident keys or aliases can provide for authentication based on decryption operations instead of producing a digital signature.

The DSC responds to requests from an external entity to attest to the provenance and integrity of

platform characteristics contained within the DSC.

Integrity reporting is the process of attesting to platform characteristics (including those recorded in status registers in a DSC). The philosophy behind integrity measurement, logging, and reporting is that a platform may enter any state possible—including undesirable or insecure states—but can still accurately report measurements derived from data attributing to those states. In this context, data can include, but is not limited to, code, security-critical configurations, values of registers, including status registers. An independent process may evaluate the integrity states and determine an appropriate response.

A.3. Flaw Remediation

The following SARs for ALC_FLR are purely optional and are not required to be added to any ST conformant to this collaborative Protection Profile. If the ST author decides to add ALC_FLR to the ST, only one out of the following SAR components shall be selected.

A.3.1. ALC_FLR.1 Basic flaw remediation

This component is targeted at the flaw remediation procedures applied by the developer to ensure that all reported security flaws in each release of the TOE are tracked and corrected. The evaluator performs the CEM work units associated with ALC_FLR.1.

A.3.2. ALC_FLR.2 Flaw reporting procedures

This component is targeted at the flaw remediation procedures applied by the developer to ensure that all reported security flaws in each release of the TOE are tracked and corrected. In addition, the developer's flaw remediation guidance is analysed to ensure that users are aware how to correctly report security flaws to the developer. The evaluator performs the CEM work units associated with ALC_FLR.2.

A.3.3. ALC_FLR.3 Systematic flaw remediation

This component is targeted at the flaw remediation procedures applied by the developer to ensure that all reported security flaws in each release of the TOE are tracked and corrected. In addition, the developer's flaw remediation guidance is analysed to ensure that users are aware how to correctly report security flaws to the developer. Flaw remediation procedures of the developer need to describe how users can register to receive flaw reports and corrections. The procedures also need to ensure timely responses to reports of security flaws and automatic distribution of security flaw reports. The evaluator performs the CEM work units associated with ALC_FLR.3.

Appendix B: Selection-Based Requirements

As indicated in the introduction to this cPP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this cPP. There are additional requirements based on selections in the body of the cPP: if certain selections are made, then additional requirements below will need to be included.

B.1. Cryptographic Support

B.1.1. FCS_CKM.1/AKG Cryptographic Key Generation - Asymmetric Key

FCS_CKM.1/AKG Cryptographic Key Generation - Asymmetric Key

FCS_CKM.1.1/AKG

The TSF shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [**selection**: *cryptographic key generation algorithm*] and specified cryptographic **algorithm parameters** [**selection**: *cryptographic algorithm parameters*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_CKM.1/AKG:

Table 15. Allowed choices for FCS_CKM.1/AKG

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
RSA	RSA	Modulus of size [selection: 2048 bit, 3072 bit]	NIST FIPS PUB 186-5 (Section A.1.1)
ECC-ERB	ECC - Extra Random Bits	Elliptic Curve [selection: P-256, brainpoolP256r1, P-384, brainpoolP384r1, P-521, brainpoolP512r1]	NIST FIPS PUB 186-5 (Section A.2.1) [selection: NIST SP 800-186 (Section 3) [NIST Curves], RFC 5639 (Section 3) [Brainpool curves]]
ECC-RS	ECC - Rejection Sampling	Elliptic Curve [selection: P-256, brainpoolP256r1, P-384, brainpoolP384r1, P-521, brainpoolP512r1]	NIST FIPS PUB 186-5 (Section A.2.2) [selection: NIST SP 800-186 (Section 3) [NIST Curves], RFC 5639 (Section 3) [Brainpool curves]]

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
FFC-ERB	FFC - Extra Random Bits	Static domain parameters approved for [selection: IKE groups [selection: MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [selection: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.3) [key pair generation] [selection: RFC 3526 [IKE groups], RFC 7919 [TLS groups]]
FFC-RSA	FFC - Rejection Sampling	Static domain parameters approved for [selection: IKE groups [selection: MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [selection: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]	NIST SP 800-56A Revision 3 (Section 5.6.1.1.4) [key pair generation] [selection: RFC 3526 [IKE groups], RFC 7919 [TLS groups]]
EdDSA	EdDSA	Domain parameters approved for elliptic curves [selection: Edwards25519, Edwards448]	FIPS PUB 186-5 (Section 6.2.1) [key-pair generation] NIST SP 800-186 (Section 3.2.3) [Edwards Curves]
KCDSA	KCDSA	Domain parameters generation with (L, N) = [selection: (2048, 224), (2048, 256), (3072, 256)] bits	ISO/IEC 14888-3:2018 (Subclause 6.3) [KCDSA]
EC-KCDSA	EC-KCDSA	Elliptic Curves [selection: P-224, B-233, K-233, P-256, B-283, K-283]	ISO/IEC 14888-3:2018 (Subclause 6.7) [EC-KCDSA] NIST SP 800-186 (Section 3) [NIST Curves]
LMS	LMS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], Winternitz parameter = [selection: 1, 2, 4, 8], and tree height [selection: 5, 10, 15, 20, 25]	RFC 8554 [LMS] NIST SP 800-208 [parameters]

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Algorithm Parameters	List of Standards
HSS	HSS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], Winternitz parameter = [selection: 1, 2, 4, 8], tree height [selection: 5, 10, 15, 20, 25], and number of levels = [selection: 1, 2, 3, 4, 5, 6, 7, 8]	RFC 8554 [HSS] NIST SP 800-208 [parameters]
XMSS	XMSS	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], tree height [selection: 10, 16, 20]	RFC 8391 [XMSS] NIST SP 800-208 [parameters]
XMSS^{MT}	XMSS^{MT}	Private key size = [selection: 192 bits with [selection: SHA-256/192, SHAKE256/192], 256 bits with [selection: SHA-256, SHAKE256]], (total tree height, number of levels) = [selection: (20, 2), (20, 4), (40, 2), (40, 4), (40, 8), (60, 3), (60, 6), (60, 12)]	RFC 8391 [XMSS^{MT}] NIST SP 800-208 [parameters]
ML-KEM	ML-KEM	Parameter set = [selection: ML-KEM-512, ML-KEM-768, ML-KEM-1024]	NIST FIPS 203 (Section 7.1)
ML-DSA	ML-DSA	Parameter set = [selection: ML-DSA-44, ML-DSA-65, ML-DSA-87]	NIST FIPS 204 (Section 5.1)

Application Note 46

For RSA the choice of the modulus implies the resulting key sizes of the public and private keys generated using the specified standard methods.

Finite Field Cryptography (FFC) DSA may only be used for verifying old digital signatures and time stamps, if this is explicitly allowed by the application domain. "FFC-ERB" or "FFC-RS" may be claimed only for generating private and public keys when "DH" is claimed in FCS_CKM_EXT.7.

When generating ECC keys pairs for key agreement and if "ECDH" or "ECDH-Ed" is claimed in FCS_CKM_EXT.7, then "ECC-ERB" or "ECC-RS" is also claimed. The sizes of the private key, which is a scalar, and the public key, which is a point on the elliptic curve, are determined by the choice of the curve.

When generating ECC key pairs for digital signature generation and if "ECDSA" or "EC-KCDSA" are claimed in FCS_COP.1/SigGen, then "ECC-ERB" or "ECC-RS" is also claimed. The sizes of the private key, which is a scalar, and the public key, which is a point on the elliptic curve, are determined by the choice of the curve.

When generating EdDSA key pairs for digital signatures and if "EdDSA" is claimed in FCS_COP.1/SigGen, then "EdDSA" is claimed here. The chosen domain parameters determine the size of the private keys and the public keys.

For LMS, HSS, XMSS, and XMSS^{MT}, the key sizes do not represent the expected security strength. All key sizes given here correspond to an expected security strength of 128 bits, per NIST SP 800-208.

For HSS and XMSS^{MT} the same hash or XOF function is used at each level. Within each level, the same Winternitz parameter is used but can be different for each level. For HSS, within each level, the same tree height is used but can be different for each level.

B.1.2. FCS_CKM.1/SKG Cryptographic Key Generation - Symmetric Key

FCS_CKM.1/SKG Cryptographic Key Generation - Symmetric Key

FCS_CKM.1.1/SKG

The TSF shall generate **symmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm [**selection**: *cryptographic key generation algorithm*] and specified cryptographic key sizes [**selection**: *cryptographic key sizes*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_CKM.1/SKG:

Table 16. Allowed choices for FCS_CKM.1/SKG

Identifier	Cryptographic Key Generation Algorithm	Cryptographic Key Sizes	List of Standards
RSK	Direct Generation from a Random Bit Generator as specified in FCS_RBG.1	[selection: 128, 192, 256, 512] bits	NIST SP 800-133 Revision 2 (Section 6.1). [Direct generation of symmetric keys]

Application Note 47

Include this requirement if the TOE supports creating symmetric keys directly from the output of an RBG without further conditioning.

To derive symmetric keys from other keying material, see FCS_CKM.5. To derive symmetric keys from passwords, see FCS_CKM_EXT.8. To derive symmetric keys from keying material contributed from two parties, see FCS_CKM_EXT.7.

See FCS_RBG.1 for requirements about appropriate entropy for selected cryptographic key sizes.

B.1.3. FCS_CKM_EXT.3 Cryptographic Key Access

FCS_CKM_EXT.3 Cryptographic Key Access

FCS_CKM_EXT.3.1

The TSF shall use specified cryptographic key access methods [selection: *key encapsulation* ***as specified in FCS_COP.1/KeyEncap***, *key wrapping* ***as specified in FCS_COP.1/KeyWrap***, *key encryption* ***as specified in FCS_COP.1/SKC or FCS_COP.1/AEAD***] to access keys when performing [selection: *cryptographic key archival, cryptographic key backup, cryptographic key escrow, cryptographic key recovery, cryptographic key import, cryptographic key export*].

B.1.4. FCS_CKM.5 Cryptographic Key Derivation

FCS_CKM.5 Cryptographic Key Derivation

FCS_CKM.5.1

The TSF shall derive cryptographic keys [**selection**: *key type*] from [**selection**: *input parameters*], in accordance with a specified cryptographic key derivation algorithm [**selection**: *key derivation algorithm*] and specified cryptographic key sizes [**selection**: *key sizes*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_CKM.5.

Table 17. Allowed choices for FCS_CKM.5

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-CTR	Input key(s), label, context, output length	KPF2 - KDF in Counter Mode using [selection: AES-128-CMAC, AES-192-CMAC, AES-256-CMAC, Camellia-128-CMAC, Camellia-192-CMAC, Camellia-256-CMAC, CMAC-HIGHT-128, CMAC-LEA-128, CMAC-LEA-256, CMAC-SEED-128, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512] as the PRF	[selection: 128, 192, 256, 512] bits	[selection: ISO/IEC 11770-6:2016 (Subclause 7.3.2) [KPF2], NIST SP 800-108 Revision 1 Update 1 (Section 4.1) [KDF in Counter Mode]]

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-FB	Input key(s), label, context, IV output length	KPF3 - KDF in Feedback Mode using [selection: AES-128-CMAC, AES-192-CMAC, AES-256-CMAC, Camellia-128-CMAC, Camellia-192-CMAC, Camellia-256-CMAC, CMAC-HIGHT-128, CMAC-LEA-128, CMAC-LEA-256, CMAC-SEED-128, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512] as the PRF	[selection: 128, 192, 256, 512] bits	[selection: ISO/IEC 11770-6:2016 (Subclause 7.3.3) [KPF3], NIST SP 800-108 Revision 1 Update 1 (Section 4.2) [KDF in Feedback Mode]]
KDF-DPI	Input key(s), label, context, output length	KPF4 - KDF in Double-Pipeline Iteration Mode using [selection: AES-128-CMAC, AES-192-CMAC, AES-256-CMAC, Camellia-128-CMAC, Camellia-192-CMAC, Camellia-256-CMAC, CMAC-HIGHT-128, CMAC-LEA-128, CMAC-LEA-256, CMAC-SEED-128, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512] as the PRF	[selection: 128, 192, 256, 512] bits	[selection: ISO/IEC 11770-6:2016 (Subclause 7.3.4) [KPF3], NIST SP 800-108 Revision 1 Update 1 (Section 4.3) [KDF in Double-Pipeline Iteration Mode]]
KDF-KMAC	Input key(s), label, context, output length	[selection: KMAC128, KMAC256]	[selection: 128, 192, 256, 512] bits	NIST SP 800-108 Revision 1 Update 1 (Section 4.4 "KDF Using KMAC")
KDF-XOR	More than one intermediary key	exclusive OR (XOR)	[selection: 128, 192, 256, 512] bits	N/A
KDF-ENC	Two intermediary keys	Encrypting using an algorithm specified in [selection: FCS_COP.1/AEAD, FCS_COP.1/SKC]	[selection: 128, 192, 256, 512] bits	N/A

Key Type	Input Parameters	Key Derivation Algorithm	Key Sizes	List of Standards
KDF-HASH	Shared secret	Hash function from FCS_COP.1/Hash	[selection: 128, 192, 256, 512] bits	NIST SP 800-56C Revision 2 (Section 4.1, Option 1) [One-Step Key Derivation]
KDF-MAC-1S	Shared secret, salt, output length, fixed information	Keyed Hash function from FCS_COP.1/KeyedHash	[selection: 128, 192, 256, 512] bits	NIST SP 800-56C Revision 2 (Section 4.1, Options 2, 3) [One-Step Key Derivation]
KDF-MAC-2S	Shared secret, salt, IV, output length, fixed information	MAC Step [selection: AES-128-CMAC, AES-192-CMAC, AES-256-CMAC, Camellia-128-CMAC, Camellia-192-CMAC, Camellia-256-CMAC, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512] as the MAC and; KDF Step [selection: KDF-CTR, KDF-FB, KDF-DPI] using [selection: AES-128-CMAC, AES-192-CMAC, AES-256-CMAC, Camellia-128-CMAC, Camellia-192-CMAC, Camellia-256-CMAC, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-512] as PRF	[selection: 128, 192, 256, 512] bits	NIST SP 800-56C Revision 2 (Section 5) [Two-Step Key Derivation]

Application Note 48

In KDF-MAC-2S, if AES-N-CMAC or Camellia-N-CMAC is selected in the MAC step, then select CMAC with the same cipher and a 128-bit key in the KDF step, and select 128 as the output key size. If HMAC is selected in the MAC step, then select the same HMAC in the KDF.

B.1.5. FCS_CKM_EXT.7 Cryptographic Key Agreement

FCS_CKM_EXT.7 Cryptographic Key Agreement

FCS_CKM_EXT.7.1

The TSF shall derive shared cryptographic keys with input from multiple parties in accordance with specified cryptographic key agreement algorithms [**selection**: *cryptographic algorithm*] and specified cryptographic parameters [**selection**: *cryptographic algorithm parameters*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_CKM_EXT.7.

Table 18. Allowed choices for FCS_CKM_EXT.7

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
KAS2	RSA	Modulus of size [selection: 2048, 3072, 4096, 6144, 8192] bits	NIST SP 800-56B Revision 2 (Section 8.3) [KAS2]
DH	Finite Field Cryptography Diffie-Hellman	Static domain parameters approved for [selection: IKE groups [selection: MODP-2048, MODP-3072, MODP-4096, MODP-6144, MODP-8192], TLS groups [selection: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192]]	NIST SP 800-56A Revision 3 (Section 5.7.1.1) [DH] [selection: RFC 3526 [IKE Groups], RFC 7919 [TLS Groups]]
ECDH	Elliptic Curve Diffie-Hellman	Elliptic Curve [selection: P-256, brainpoolP256r1, P-384, brainpoolP384r1, P-521, brainpoolP512r1]	NIST SP 800-56A Revision 3 (Section 5.7.1.2) [ECDH] [selection: NIST SP 800-186 (Section 3.2.1) [NIST Curves], RFC 5639 (Section 3) [Brainpool curves]]
ECDH-Ed	ECDH with Montgomery Curves	Domain parameters approved for elliptic curves [selection: curve25519, curve448]	RFC 7748 (Section 5) [ECDH-Ed] NIST SP 800-186 (Section 3.2.2) [Montgomery Curves]

Application Note 49

ST authors should consider the assumptions that opposite parties in the operational environment contribute keying material that meets the same requirements.

B.1.6. FCS_CKM_EXT.8 Password-Based Key Derivation

FCS_CKM_EXT.8 Password-Based Key Derivation

FCS_CKM_EXT.8.1

The TSF shall perform password-based key derivation functions in accordance with a specified cryptographic algorithm HMAC-[selection: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*], with iteration count of [assignment: *number of iterations*] using a randomly generated salt of length [assignment: *equal to or greater than 128*] and output cryptographic key sizes [selection: *128, 192, 256, [assignment: greater than 128]*] bits that meet the following standard: NIST SP 800-132 Section 5.3 (PBKDF2).

Application Note 50

Password-based key derivation is different from regular key derivation in that passwords have very limited entropy. As a result, additional constraints, work, or entropy to achieve acceptable levels of security when using password-based key derivation algorithms are needed. This component only adds work through increased iterations and use of salts; it does not consider additional constraints or entropy.

The TSF is allowed to use PBKDF2 to condition passwords in the context of password-based authentication. In this scenario, the output of PBKDF2 is not directly used as a cryptographic key, but only stored as a reference value (commonly called "password hash") to compare against when performing authentication. The "cryptographic key size" selected in this element corresponds to the length of the password hash.

NIST recommends a minimum "number of iterations" of 1000 but prefers the largest number feasible given performance constraints.

NIST recommends that the randomly generated portion of the salt have length of at least 128 bits and is derived from a Random Bit Generation. Therefore FCS_CKM_EXT.8 depends on FCS_OTV_EXT.1.

B.1.7. FCS_COP.1/AEAD Cryptographic Operation - Authenticated Encryption with Associated Data

FCS_COP.1/AEAD Cryptographic Operation - Authenticated Encryption with Associated Data

FCS_COP.1.1/AEAD

The TSF shall perform [*authenticated encryption with associated data*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic key sizes [**selection**: *cryptographic key sizes*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/AEAD.

Table 19. Allowed choices for FCS_COP.1/AEAD

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CCM	AES in CCM mode with non-repeating nonce, minimum size of 64 bits	[selection: 128, 192, 256] bits	[selection: ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [selection: ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C] [CCM]
AES-GCM	AES in GCM mode with non-repeating IVs using [selection: deterministic, RBG-based] IV construction; the tag must be of length [selection: 96, 104, 112, 120, or 128] bits	[selection: 128, 192, 256], bits	[selection: ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [selection: ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D] [GCM]
CAM-CCM	Camellia in CCM mode with non-repeating nonce, minimum size of 64 bits	[selection: 128, 192, 256] bits	ISO/IEC 18033-3:2010 (Subclause 5.3) [Camellia] [selection: ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C] [CCM]
CAM-GCM	Camellia in GCM mode with non-repeating IVs using [selection: deterministic, RBG-based] IV construction; the tag must be of length [selection: 96, 104, 112, 120, or 128] bits	[selection: 128, 192, 256] bits	ISO/IEC 18033-3:2010 (Subclause 5.3) [Camellia] [selection: ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D] [GCM]
SEED-CCM	SEED in CCM mode with non-repeating nonce, minimum size of 64 bits	128 bits	ISO/IEC 18033-3:2010 (Subclause 5.4) [SEED] [selection: ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C] [CCM]
SEED-GCM	SEED in GCM mode with non-repeating IVs using [selection: deterministic, RBG-based] IV construction; the tag must be of length [selection: 96, 104, 112, 120, or 128] bits	128 bits	ISO/IEC 18033-3:2010 (Subclause 5.4) [SEED] [selection: ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D] [GCM]

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
LEA-CCM	LEA in CCM mode with non-repeating nonce, minimum size of 64 bits	[selection: 128, 192, 256] bits	ISO/IEC 29192-2:2019 (Subclause 6.3) [LEA] [selection: ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C] [CCM]
LEA-GCM	LEA in GCM mode with non-repeating IVs using [selection: deterministic, RBG-based] IV construction; the tag must be of length [selection: 96, 104, 112, 120, or 128] bits	[selection: 128, 192, 256] bits	ISO/IEC 29192-2:2019 (Subclause 6.3) [LEA] [selection: ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38D] [GCM]

B.1.8. FCS_COP.1/CMAC Cryptographic Operation - CMAC

FCS_COP.1/CMAC Cryptographic Operation - CMAC

FCS_COP.1.1/CMAC

The TSF shall perform [*CMAC*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic key sizes [**selection**: *cryptographic key sizes*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/CMAC.

Table 20. Allowed choices for FCS_COP.1/CMAC

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
AES-CMAC	AES using CMAC mode	[selection: 128, 192, 256] bits	[selection: ISO/IEC 18033-3:2010 (Subclause 5.2), FIPS PUB 197] [AES] [selection: ISO/IEC 9797-1:2011 Subclause 7.6, NIST SP 800-38B] [CMAC]
CAM-CMAC	Camellia using CMAC mode	[selection: 128, 192, 256] bits	ISO/IEC 18033-3:2010 Subclause 5.3 [Camellia] [selection: ISO/IEC 9797-1:2011 Subclause 7.6, NIST SP 800-38B] [CMAC]

B.1.9. FCS_COP.1/KeyEncap Cryptographic Operation - Key Encapsulation

FCS_COP.1/KeyEncap Cryptographic Operation - Key Encapsulation

FCS_COP.1.1/KeyEncap

The TSF shall perform [*key encapsulation*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic **algorithm parameters** [**selection**: *cryptographic algorithm parameters*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/KeyEncap.

Table 21. Allowed choices for FCS_COP.1/KeyEncap

Identifier	Cryptographic Algorithm	Cryptographic Algorithm Parameters	List of Standards
KAS1	RSASVE.Generate & RSASVE.Recover [RSA-single party]	Key size = [selection: 2048, 3072, 4096, 8192] bits	NIST SP 800-56B Revision 2 (Section 7.2.1.2 & 7.2.1.3)
KTS-OAEP	RSA-OAEP.Encrypt & RSA-OAEP.Decrypt [RSA-OAEP]	Key size = [selection: 2048, 3072, 4096, 8192] bits	NIST SP 800-56B Revision 2 (Section 7.2.2.3 & 7.2.2.4)
ML-KEM	ML-KEM.Encaps & ML-KEM.Decaps [ML-KEM]	Parameter set = [selection: ML-KEM-512, ML-KEM-768, ML-KEM-1024]	NIST FIPS 203 (Section 7.2)

B.1.10. FCS_COP.1/KeyWrap Cryptographic Operation - Key Wrapping

FCS_COP.1/KeyWrap Cryptographic Operation - Key Wrapping

FCS_COP.1.1/KeyWrap

The TSF shall perform [*key wrapping*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic **algorithm parameters** [**selection**: *cryptographic algorithm parameters*] that meet the following: [**selection**: *list of standards*].

Table 22. Allowed choices for FCS_COP.1/KeyWrap

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
KW	[selection: AES, CAM, SEED, LEA] in KW mode	[selection: (AES, CAM, SEED, LEA) 128, (AES, CAM, LEA) 192, (AES, CAM, LEA) 256] bits	[selection: ISO/IEC 19772:2020 (Clause 6), NIST SP 800-38F (Section 6.2)] [KW mode]
KWP	[selection: AES, CAM, SEED, LEA] in KWP mode	[selection: (AES, CAM, SEED, LEA) 128, (AES, CAM, LEA) 192, (AES, CAM, LEA) 256] bits	NIST SP 800-38F (Section 6.3) [KWP mode]
CCM	[selection: AES, CAM, LEA, SEED] in CCM mode with non-repeating nonce, minimum size of 64 bits	[selection: (AES, CAM, SEED, LEA) 128, (AES, CAM, LEA) 192, (AES, CAM, LEA) 256] bits	[selection: ISO/IEC 19772:2020 (Clause 7), NIST SP 800-38C] [CCM mode]

Identifier	Cryptographic Algorithm	Cryptographic Key Sizes	List of Standards
GCM	[selection: AES, CAM, LEA, SEED] in GCM mode with non-repeating IVs 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 t must be one of the values 96, 104, 112, 120, and 128 bits.	[selection: (AES, CAM, SEED, LEA) 128, (AES, CAM, LEA) 192, (AES, CAM, LEA) 256] bits	[selection: ISO/IEC 19772:2020 (Clause 10), NIST SP 800-38C] [CCM mode]

B.1.11. FCS_COP.1/XOF Extendable-Output Function

FCS_COP.1/XOF Extendable-Output Function

FCS_COP.1.1/XOF

The TSF shall perform [*extendable-output function*] in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and **parameters** [**selection**: *parameters*] that meet the following: [**selection**: *list of standards*].

The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/XOF.

Table 23. Allowed choices for FCS_COP.1/XOF

Cryptographic Algorithm	Parameters	List of Standards
cSHAKE	Output length d = [selection: 128, 256] bits and function [selection: SHAKEd, KECCAK[2d]]	NIST SP 800-185 Section 3 [cSHAKE], Section 6.2 [SHAKE] NIST FIPS PUB 202 Section 5 [KECCAK]
KMACXOF	Output length d = [selection: 128, 256] bits	NIST SP 800-185 Section 4.3.1 [KMACXOF]
SHAKE	Output length d = [selection: 128, 256] bits	NIST FIPS PUB 202 Section 6.2 [SHAKE]

Application Note 51

The functions in cSHAKE depend on the output length d., i.e. SHAKEd is either SHAKE128 for d = 128 or SHAKE256 for d = 256. Similarly, KECCAK[2d] is either KECCAK[256] for d = 128 or KECCAK[512] for d = 256.

Note that KECCAK is a cryptographic primitive which is not expected to have a direct interface exposed to the user of the TOE.

B.2. User Data Protection

B.2.1. FDP_DAU.1/Prove Basic Data Authentication (for Use with the Prove Service)

FDP_DAU.1/Prove Basic Data Authentication (for Use with the Prove Service)

FDP_DAU.1.1/Prove

The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of [**selection**: *[assignment: list of objects or information types] declared valid by the TSF, [assignment: list of objects or information types] declared valid by an authenticated user*].

FDP_DAU.1.2/Prove

The TSF shall provide [assignment: *list of subjects*] with the ability to verify evidence of the validity of the indicated information.

Application Note 52

This SFR describes the output of the Prove service provided by the DSC. The evidence of validity or authenticity, or other evidence derived, is expected to be processed by the RoT for Measurement. Additionally, the use of a RoT for Reporting presupposes a logging capability or other means of generating state information that could be conveyed to external entities. Therefore, FDP_DAU.1.1/Prove is claimed if-and-only-if Root of Trust for Measurement and Root of Trust for Reporting are both selected in FPT_ROT_EXT.1.1. An 'authenticated user' in the sense of the selection in FDP_DAU.1.1/Prove means a user who has been authenticated by the DSC according to the mechanisms of FIA_UAU.5.

In FDP_DAU.1.1/Prove, the DSC will issue a validity-stamped or authenticity-stamped piece of data. In this case, validity-stamped means that the form of the issued data enables an external entity to verify that the data has been issued via the DSC's Prove service. The implementation might be via a DSC cryptographic signature, or a MAC using a symmetric key shared with the receiver, for example. Authenticity-stamped means that the receiver of the data can verify that the user providing this data is authentic.

Data that would need to be validity-stamped includes data over which the DSC is the authority, such as the state of its own firmware. Data that would need to be authenticity-stamped includes data about which the DSC knows nothing, but where it will issue the data with a statement that the DSC has authenticated the source of this data.

For data that is validity-stamped, the DSC does nothing but respond to a request to issue the data; thus, authentication of the user issuing the data is not needed and is covered by FDP_DAU.1/Prove. Otherwise, in the case the DSC has no understanding of this data, a step is needed via FIA_UAU.5 by which the DSC authenticates the user for this service, and that the DSC or Prove service will therefore vouch for the user, not the validity of the data itself.

B.2.2. FDP_FRS_EXT.2 Factory Reset Behavior

FDP_FRS_EXT.2 Factory Reset Behavior

FDP_FRS_EXT.2.1

Upon initiation of a factory reset, the TSF shall destroy [*all non-permanent SDEs and SDOs*] and restore the following **pre-installed SDOs** to their factory settings: [*assignment: **pre-installed SDOs to be** restored by a factory reset*].

Application Note 53

Not all DSCs permit factory reset functionality. Those that do are expected to perform a factory reset in a manner that prevents any inadvertent disclosure of security-relevant data that was present on the DSC prior to the factory reset. For DSCs that permit factory reset functionality (as indicated by selection of factory reset the TOE wiping out all non-permanent SDEs and SDOs, as described by FDP_FRS_EXT.2 in FIA_AFL_EXT.1.3, or by no actions or conditions NOT being selected in FDP_FRS_EXT.1.1), this SFR must be claimed.

B.3. Identification and Authentication

B.3.1. FIA_AFL_EXT.2 Authorization Failure Response

FIA_AFL_EXT.2 Authorization Failure Response

FIA_AFL_EXT.2.1

When the TSF locks an **SDO** (i.e. prevents authorization attempts for an **SDO**) due to a user exceeding the allowed threshold for unsuccessful authorization attempts, then only an administrator may unlock access to the **SDO** and reset the corresponding failed authorization attempt counter.

Application Note 54

This SFR is applicable only when the TSF's response to excessive authorization failures selects prevent all future authorization attempts indefinitely (i.e., lock), as described by FIA_AFL_EXT.2 as specified by FIA_AFL_EXT.1.3.

B.4. Protection of the TSF

B.4.1. FPT_FLS.1/FW Failure with Preservation of Secure State (Firmware)

FPT_FLS.1/FW

Failure with Preservation of Secure State (Firmware)

FPT_FLS.1.1/FW

The TSF shall preserve a secure state when the following types of **firmware** failures occur: [*authenticity violation, integrity violation, rollback violation*].

Application Note 55

A DSC's ability to handle failures related to authenticity, integrity, and invalid versions of firmware is not applicable for immutable firmware.

The phrase "secure state" refers to a state in which the TOE has consistent TSF data and a TSF that can correctly enforce the policy. The TOE ensures that no further processing of TSF or user data

takes place while in an insecure state. This state may be the initial "boot" of a clean system, or it might be some check-pointed state. It is expected that in most cases, the TOE will halt and require a reset or re-initialization to return to a known secure state.

B.4.2. FPT_MFW_EXT.2 Basic Firmware Integrity

FPT_MFW_EXT.2 Basic Firmware Integrity

FPT_MFW_EXT.2.1

The TSF shall have the ability to verify the integrity of the firmware.

FPT_MFW_EXT.2.2

The TSF shall provide a capability to generate evidence of the integrity of the firmware.

Application Note 56

Data and firmware integrity is not applicable for immutable firmware.

The TOE guarantees the integrity of the firmware by verifying its integrity.

Verifying the integrity of the firmware could be accomplished by guaranteeing the validity of firmware within the TOE prior to execution.

This requirement covers the case of ensuring the firmware is trustworthy in immutable form or mutable through any firmware updates, since the integrity and authenticity are checked prior to execution.

FCS_COP.1/SigVer applies if the TOE provides the capability to update the TOE firmware and uses digital signatures for update verification. FCS_COP.1/CMAC or FCS_COP.1/KeyedHash applies if the TOE provides the capability to update the TOE firmware and uses MAC verification for update verification. The ST author should choose the algorithm implemented to perform digital signatures or MAC verification. For the algorithms chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.

B.4.3. FPT_MFW_EXT.3 Firmware Authentication with Identity of Guarantor

FPT_MFW_EXT.3 Firmware Authentication with Identity of Guarantor

FPT_MFW_EXT.3.1

The TSF shall have the ability to verify the authenticity of the firmware.

FPT_MFW_EXT.3.2

The TSF shall provide a capability to generate evidence of the authenticity of the firmware.

Application Note 57

Firmware authentication is not applicable for immutable firmware.

The TOE guarantees the authenticity of the firmware by verifying its signature.

Verifying the authenticity of the firmware could be accomplished by guaranteeing the validity of firmware within the TOE prior to execution.

This requirement covers the case of ensuring the firmware is trustworthy in immutable form or mutable through any firmware updates, since the integrity and authenticity are checked prior to execution.

FCS_COP.1/SigVer applies if the TOE provides the capability to update the TOE firmware and uses digital signatures for update verification. The ST author should choose the algorithm implemented to perform digital signatures. For the algorithms chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.

B.4.4. FPT_RPL.1/Rollback Replay Detection (Rollback)

FPT_RPL.1/Rollback Replay Detection (Rollback)

FPT_RPL.1.1/Rollback

The TSF shall detect replay for the following entities: [*previous firmware builds*].

FPT_RPL.1.2/Rollback

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

Application Note 58

The TSF data is used as a guarantee of the ordinal identifier of the firmware instance. When a firmware load is requested, the TSF ensures the authenticated firmware ordinal identifier is greater than or equal to the previously authenticated firmware identifier. For example, this could be accomplished by ensuring the validated instance of the firmware to be loaded is greater than or equal to the instance previously validated and loaded into the TOE. By loading a previous instance of firmware, it potentially opens up the device to known vulnerabilities.

B.4.5. FPT_STM_EXT.1 Reliable Time Counting

FPT_STM_EXT.1 Reliable Time Counting

FPT_STM_EXT.1.1

The TSF shall be able to provide a reliable [selection: *internal time stamp, external time stamp, monotonically increasing counter*] to measure the passage of time.

Application Note 59

It is acceptable for the TSF to provide timestamp data either through an internal clock or a counter. It is also permissible for the TSF to obtain time data from a clock contained within the same physical enclosure in which the TOE is embedded (e.g. a mobile device).

B.5. Trusted Path/Channels

B.5.1. FTP_ITC_EXT.1 Cryptographically Protected Communications Channels

FTP_ITC_EXT.1 Cryptographically Protected Communications Channels

FTP_ITC_EXT.1.1

The TSF shall use [*assignment: cryptographic protocol*] to provide a communication channel between itself and [*assignment: list of entities external to the TOE*] that protects channel data from disclosure and ensures the integrity of channel data.

Application Note 60

Entities external to the TOE include applications that communicate with the TOE such as authentication capabilities (e.g. biometrics reader), external storage, and interfaces with an external DSC.

B.5.2. FTP_ITE_EXT.1 Encrypted Data Communications

FTP_ITE_EXT.1 Encrypted Data Communications

FTP_ITE_EXT.1.1

The TSF shall encrypt data for transfer between the TOE and [*assignment: list of entities external to the TOE*] using a cryptographic algorithm and key size as specified in FCS_COP.1/**SKC**, and using [**selection**:

- *Pre-shared keys;*
- *Key agreement according to FCS_CKM_EXT.7;*
- *Key encapsulation according to FCS_COP.1/KeyEncap;*
- *Keys exchanged using a physically protected communication mechanism conformant with FTP_ITP_EXT.1*].

Application Note 61

This requirement applies to encrypted data communications between the TOE and external entities that do not use a physically protected mechanism conforming to FTP_ITP_EXT.1, or a cryptographically protected data channel as conforming to FTP_ITC_EXT.1. For example, if data is transferred through encrypted buffers (or blobs) then this requirement applies. If data is transferred through a physically protected channel, then FTP_ITP_EXT.1 applies. This requirement would apply, for example, for communications implemented through a shared data buffer.

B.5.3. FTP_ITP_EXT.1 Physically Protected Channel

FTP_ITP_EXT.1 Physically Protected Channel

FTP_ITP_EXT.1.1

The TSF shall provide a physically protected communication channel between itself and [*assignment: list of other IT entities within the same platform*].

Appendix C: Extended Component Definitions

This appendix contains the definitions for the extended requirements that are used in the cPP, including those used in Appendix A and Appendix B.

(Note: formatting conventions for selections and assignments in this Appendix are those in [CC2].)

This Appendix provides a definition for all of the extended components introduced in this PP-Module. These components are identified in the following table:

Table 24. Extended Components Definitions

Functional Class	Functional Components
Cryptographic Support (FCS)	FCS_CKM_EXT Cryptographic Key Management
	FCS_OTV_EXT One-Time Value
	FCS_STG_EXT Cryptographic Key Storage
User Data Protection (FDP)	FDP_ETC_EXT Export from the TOE
	FDP_FRS_EXT Factory Reset
	FDP_ITC_EXT Import from Outside of the TOE
Identification and Authentication (FIA)	FIA_AFL_EXT Authorization Failure Handling
Security Management (FMT)	FMT_MOF_EXT Management of Functions in TSF
Protection of the TSF (FPT)	FPT_MFW_EXT Mutable/Immutable Firmware
	FPT_MOD_EXT Debug Modes
	FPT_PRO_EXT Root of Trust
	FPT_ROT_EXT Root of Trust Services
	FPT_STM_EXT Reliable Time Counting
Trusted Path/Channels (FTP)	FTP_ITC_EXT Inter-TSF Trusted Channel
	FTP_ITE_EXT Encrypted Data Communications
	FTP_ITP_EXT Physically Protected Channel

C.1. Class FCS: Cryptographic Support

C.1.1. FCS_CKM_EXT Cryptographic Key Management

Family Behavior

This family defines requirements for key life cycle operations.

Component Leveling

FCS_CKM_EXT.3 The cryptographic key access applies primarily to the storage of keys for future use and retrieval of keys for immediate use by the TOE. There may be some overlap in primitives used in other SFRs, but the end goals here are to protect the confidentiality and authenticity of the keys while in storage.

FCS_CKM_EXT.7 This SFR contains methods for multi-party key agreement in which two or more parties contribute material used to derive the shared key used by each party to encrypt and decrypt incoming and outgoing messages. TOEs can use the keys as symmetric keys, keyed-hash keys, or cryptographic keys for key derivation functions.

FCS_CKM_EXT.8 Password key derivation is different from regular key derivation since for key derivation it is expected that the input parameters have full entropy compared to the expected key strength and the passwords for password-based key derivation have limited entropy. One must add additional constraints, work, or entropy to increase the security of password-based key derivation algorithms that suffer from a lack of entropy induced by passwords sourced by human memory. These password-based key derivation algorithms should result in derived keys that have sufficient security.

Management: FCS_CKM_EXT.3, FCS_CKM_EXT.7, FCS_CKM_EXT.8

No specific management functions are identified.

Audit: FCS_CKM_EXT.3, FCS_CKM_EXT.7, FCS_CKM_EXT.8

There are no auditable events foreseen.

FCS_CKM_EXT.3 Cryptographic Key Access

Hierarchical to	No other components.
Dependencies	[FCS_CKM.1 Cryptographic key generation, or FCS_CKM.5 Cryptographic key derivation, or FCS_CKM_EXT.8 Password-based key derivation] FCS_CKM.6 Timing and event of cryptographic key destruction, [FCS_COP.1/KeyEncap Cryptographic Operation - Key Encapsulation, or FCS_COP.1/KeyWrap Cryptographic Operation - Key Wrapping, or FCS_COP.1/SKC Cryptographic Operation - Symmetric-Key Cryptography, or FCS_COP.1/AEAD Authenticated Encryption with Associated Data]

FCS_CKM_EXT.3.1

The TSF shall use specified cryptographic key access methods [selection: *key encapsulation, key wrapping, key encryption*] to access keys when performing [selection: *cryptographic key archival, cryptographic key backup, cryptographic key escrow, cryptographic key recovery, cryptographic key import, cryptographic key export*].

FCS_CKM_EXT.7 Cryptographic Key Agreement

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, or FCS_CKM.5 Cryptographic key derivation, or FCS_CKM_EXT.8 Password-based key derivation] [FCS_CKM.2 Cryptographic key distribution, or FCS_COP.1 Cryptographic operation] FCS_CKM.6 Timing and event of cryptographic key destruction

FCS_CKM_EXT.7.1

The TSF shall derive shared cryptographic keys with input from multiple parties in accordance with specified cryptographic key agreement algorithms [assignment: *cryptographic algorithm*] and specified cryptographic parameters [assignment: *cryptographic parameters*] that meet the following: [assignment: *list of standards*].

FCS_CKM_EXT.8 Password-Based Key Derivation

Hierarchical to	No other components.
Dependencies	[FCS_CKM.2 Cryptographic key distribution, or FCS_CKM_EXT.7 Cryptographic Key Agreement] FCS_COP.1/KeyedHash Cryptographic Operation - Keyed Hash FCS_CKM.6 Timing and event of cryptographic key destruction FCS_OTV_EXT.1 One-Time Value

FCS_CKM_EXT.8.1

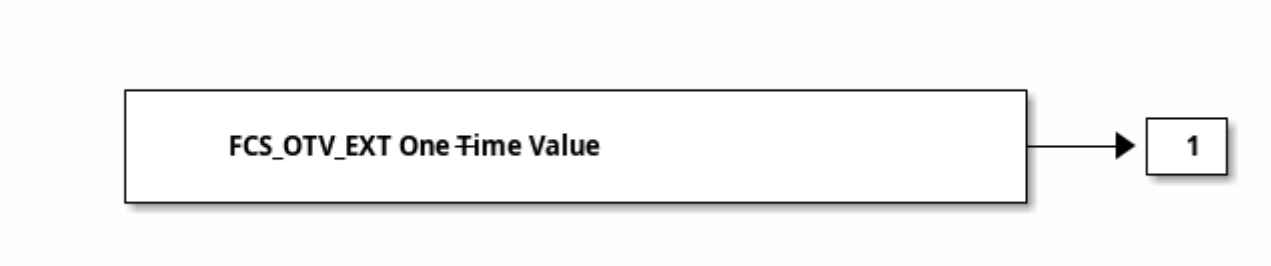
The TSF shall perform password-based key derivation functions in accordance with a specified cryptographic algorithm HMAC-[selection: *SHA-256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512*], with iteration count of [assignment: *number of iterations*] using a randomly generated salt of length [assignment: *equal to or greater than 128*] and output cryptographic key sizes [selection: *128, 192, 256, [assignment: greater than 128]*] bits that meet the following standard: NIST SP 800-132 Section 5.3 (PBKDF2).

C.1.2. FCS_OTV_EXT One-Time Value

Family Behavior

This family defines requirements for salt, nonce, and other one-time value usage.

Component Leveling



FCS_OTV_EXT.1 One-Time Value, requires that values such as salts, nonces, IVs, and initial counters be generated using random bit generation.

Management: FCS_OTV_EXT.1

No specific management functions are identified.

Audit: FCS_OTV_EXT.1

There are no auditable events foreseen.

FCS_OTV_EXT.1 One-Time Value

Hierarchical to	No other components.
Dependencies	FCS_RBG.1 Random Bit Generation (RBG)

FCS_OTV_EXT.1.1

The TSF shall perform cryptographic one-time value generation for [assignment: *algorithm or mode*] using the output of a [selection: *random bit generator as defined in FCS_RBG.1, deterministic OTV construction, [assignment: OTV construction method]*] and sizes of length that meet the following: [assignment: *list of standards*].

C.1.3. FCS_STG_EXT Cryptographic Key Storage

Family Behavior

This family defines requirements for ensuring the protection of keys and secrets.

Component Leveling

FCS_STG_EXT.1 Protected Storage, requires the TSF to enforce protected storage for keys and secrets so that they cannot be accessed or destroyed without authorization.

Management: FCS_STG_EXT.1

No specific management functions are identified.

Audit: FCS_STG_EXT.1

There are no auditable events foreseen.

FCS_STG_EXT.1 Protected Storage

Hierarchical to No other components.

Dependencies No dependencies.

FCS_STG_EXT.1.1

The TSF shall provide [assignment: *protection method*] protected storage for asymmetric private keys, symmetric keys and [selection: *persistent secrets, no other keys*].

FCS_STG_EXT.1.2

The TSF shall support the capability of [selection: *importing keys/secrets into the TOE, causing the TOE to generate keys/secrets*] upon request of [assignment: *authorized subject*].

FCS_STG_EXT.1.3

The TSF shall be capable of destroying keys/secrets in the protected storage upon request of [assignment: *authorized subject*].

FCS_STG_EXT.1.4

The TSF shall have the capability to allow only the user that [selection: *imported the key/secret, caused the key/secret to be generated*] to use the key/secret. Exceptions may only be explicitly authorized by [assignment: *authorized subject*].

FCS_STG_EXT.1.5

The TSF shall allow only the user that [selection: *imported the key/secret, caused the key/secret to be generated*] to request that the key/secret be destroyed. Exceptions may only be explicitly authorized by [assignment: *authorized subject*].

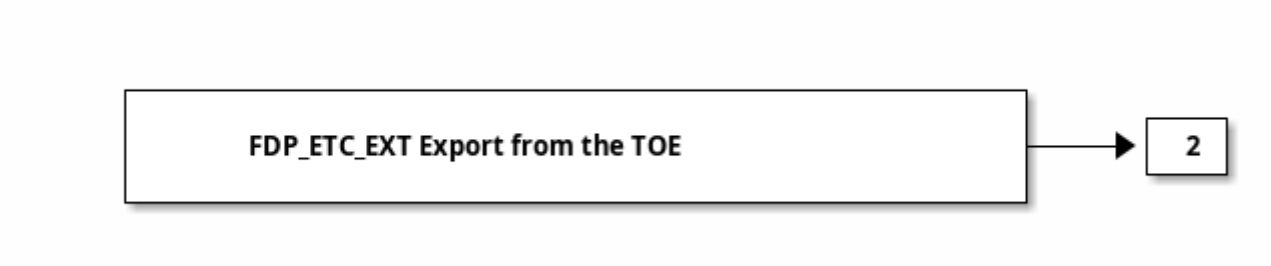
C.2. Class FDP: User Data Protection

C.2.1. FDP_ETC_EXT Export from the TOE

Family Behavior

This family defines requirements for export of TSF data outside the TOE boundary that allows for the security of that data to be maintained.

Component Leveling



FDP_ETC_EXT.2 Propagation of SDOs, requires the TSF to transmit data outside of the TOE boundary with protections applied so that they cannot be accessed by unauthorized subjects.

Management: FDP_ETC_EXT.2

No specific management functions are identified.

Audit: FDP_ETC_EXT.2

There are no auditable events foreseen.

FDP_ETC_EXT.2 Propagation of SDOs

Hierarchical to	No other components.
Dependencies	FCS_COP.1 Cryptographic Operation

FDP_ETC_EXT.2.1

The TSF shall propagate only wrapped authorization data and wrapped SDOs such that only [selection: *the TOE, authorized users*] can access them.

C.2.2. FDP_FRS_EXT Factory Reset

Family Behavior

This family defines requirements for the conditions that may trigger TOE factory reset and for identifying the data that is destroyed or restored as part of the factory reset operation.

Component Leveling

FDP_FRS_EXT.1 Factory Reset, requires the TSF to allow factory resets under specified conditions.

FDP_FRS_EXT.2 Factory Reset Behavior, requires the TSF to destroy certain TSF data and restore other TSF data after a factory reset is initiated.

Management: FDP_FRS_EXT.1

The following actions could be considered for the management functions in FMT:

- Reset TOE to factory state.

Management: FDP_FRS_EXT.2

No specific management functions are identified.

Audit: FDP_FRS_EXT.1, FDP_FRS_EXT.2

There are no auditable events foreseen.

FDP_FRS_EXT.1 Factory Reset

Hierarchical to	No other components.
Dependencies	FDP_FRS_EXT.2 Factory Reset Behavior

FDP_FRS_EXT.1.1

The TSF shall permit a factory reset of the TOE upon: [assignment: *conditions under which a factory reset is authorized*].

FDP_FRS_EXT.2 Factory Reset Behavior

Hierarchical to	No other components.
Dependencies	FDP_FRS_EXT.1 Factory Reset

FDP_FRS_EXT.2.1

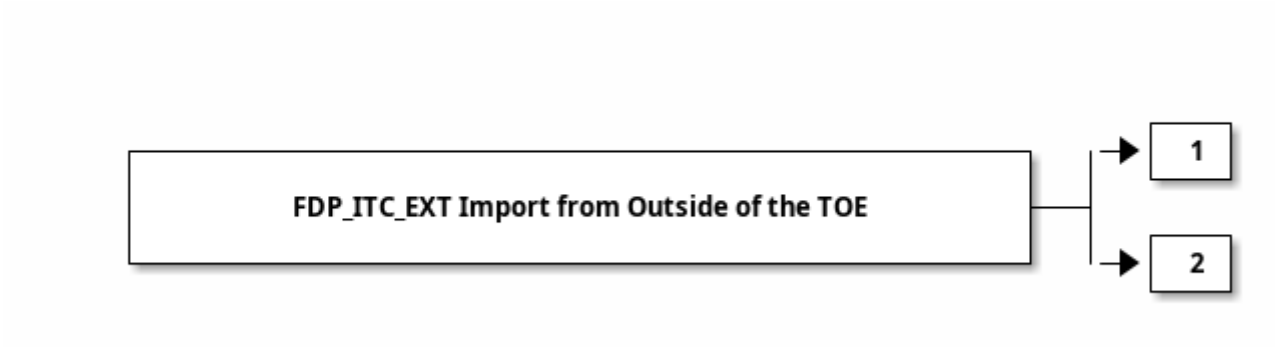
Upon initiation of a factory reset, the TSF shall destroy [assignment: *TSF data that is destroyed by factory reset*] and restore the following TSF data to their factory settings: [assignment: *TSF data that is restored by factory reset*].

C.2.3. FDP_ITC_EXT Import from Outside of the TOE

Family Behavior

This family defines requirements for handling data that is imported from outside the TOE.

Component Leveling



FDP_ITC_EXT.1 Parsing of SDEs, requires the TSF to support the import of SDEs from outside the TOE and to verify their integrity when imported.

FDP_ITC_EXT.2 Parsing of SDOs, requires the TSF to support the import of SDOs from outside the TOE and to verify their integrity when imported.

Management: FDP_ITC_EXT.1, FDP_ITC_EXT.2

No specific management functions are identified.

Audit: FDP_ITC_EXT.1, FDP_ITC_EXT.2

There are no auditable events foreseen.

FDP_ITC_EXT.1 Parsing of SDEs

Hierarchical to	No other components.
Dependencies	FCS_COP.1 Cryptographic Operation [FTP_ITC_EXT.1 Cryptographically Protected Communications Channels, FTP_ITE_EXT.1 Encrypted Data Communications, or FTP_ITP_EXT.1 Physically Protected Channel]

FDP_ITC_EXT.1.1

The TSF shall support importing SDEs using [assignment: *import method that maintains confidentiality and integrity of imported data*].

FDP_ITC_EXT.1.2

The TSF shall verify the integrity of the SDE using [assignment: *method of integrity verification*].

FDP_ITC_EXT.1.3

The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

FDP_ITC_EXT.1.4

The TSF shall bind SDEs to security attributes using [assignment: *list of ways the TSF generates security attributes and binds them to the SDEs*].

FDP_ITC_EXT.2 Parsing of SDOs

Hierarchical to	No other components.
Dependencies	FCS_COP.1 Cryptographic Operation [FTP_ITC_EXT.1 Cryptographically Protected Communications Channels, FTP_ITE_EXT.1 Encrypted Data Communications, or FTP_ITP_EXT.1 Physically Protected Channel]

FDP_ITC_EXT.2.1

The TSF shall support importing SDOs using [assignment: *import method that maintains confidentiality and integrity of imported data*].

FDP_ITC_EXT.2.2

The TSF shall verify the integrity of the SDO using [assignment: *method of integrity verification*].

FDP_ITC_EXT.2.3

The TSF shall use the security attributes associated with the imported user data.

FDP_ITC_EXT.2.4

The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC_EXT.2.5

The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

C.3. Class FIA: Identification and Authentication

C.3.1. FIA_AFL_EXT Authorization Failure Handling

Family Behavior

This family defines requirements for the TOE's behavior when repeated failed attempts to gain authorization to access TSF data occur.

Component Leveling

FIA_AFL_EXT.1 Authorization Failure Handling, requires the TSF to monitor authorization attempts, including counting and limiting the number of attempts at failed or passed authorizations.

FIA_AFL_EXT.2 Authorization Failure Response, requires the TSF to control who is authorized to unlock failed authorization attempts.

Management: FIA_AFL_EXT.1

The following actions could be considered for the management functions in FMT:

- Set authorization failure parameters.

Management: FIA_AFL_EXT.2

The following actions could be considered for the management functions in FMT:

- Unlock access to SDO following excessive failed authorization attempts.

Audit: FIA_AFL_EXT.1, FIA_AFL_EXT.2

There are no auditable events foreseen.

FIA_AFL_EXT.1 Authorization Failure Handling

Hierarchical to No other components.

Dependencies No dependencies.

FIA_AFL_EXT.1.1

The TSF shall maintain [selection: *a unique counter for [assignment: multiple separate objects each requiring authorization], one global counter covering [assignment: objects requiring authorization]*], called the failed authorization attempt counters, that counts of the number of unsuccessful authorization attempts that occur related to authorizing access to these objects.

FIA_AFL_EXT.1.2

The TSF shall maintain a [selection, choose one of: *static, administrator configurable variable*] threshold of the minimal acceptable number of unsuccessful authorization attempts that occur related to authorizing access to these objects.

FIA_AFL_EXT.1.3

When the failed authorization attempt counters [selection, choose one of: *meets, surpasses*] the threshold for unsuccessful authorization attempts, the TSF shall [assignment: *perform action that temporarily or permanently prevents access to the object*] for these objects.

FIA_AFL_EXT.1.4

The TSF shall increment the failed authorization attempt counter before it verifies the authorization.

FIA_AFL_EXT.2 Authorization Failure Response

Hierarchical to No other components.

Dependencies FIA_AFL_EXT.1 Authorization Failure Handling

FIA_AFL_EXT.2.1

When the TSF locks an object (i.e. prevents authorization attempts for an object) due to a user exceeding the allowed threshold for unsuccessful authorization attempts, then only an administrator may unlock access to the object and reset the corresponding failed authorization attempt counter.

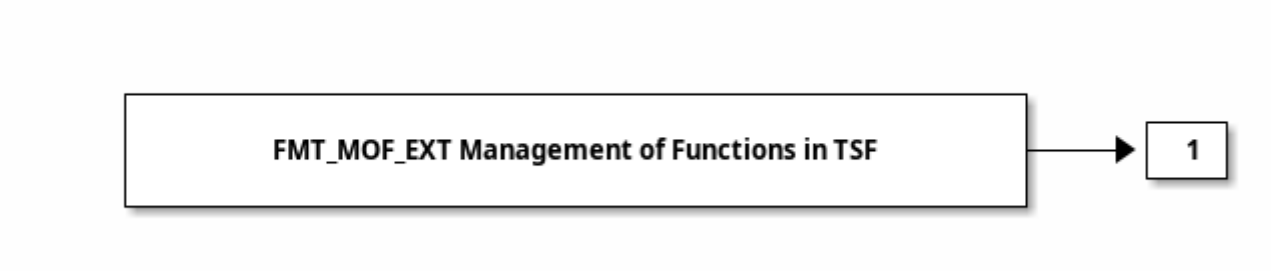
C.4. Class FMT: Security Management

C.4.1. FMT_MOF_EXT Management of Functions in TSF

Family Behavior

This family defines requirements for who is allowed to perform administrative functions.

Component Leveling



FMT_MOF_EXT.1 Management of Security Functions Behavior, requires the TSF to restrict management functionality to authorized administrators.

Management: FMT_MOF_EXT.1

No specific management functions are identified.

Audit: FMT_MOF_EXT.1

There are no auditable events foreseen.

FMT_MOF_EXT.1 Management of Security Functions Behavior

Hierarchical to	No other components.
Dependencies	FMT_SMF.1 Specification of Management Functions

FMT_MOF_EXT.1.1

The TSF shall restrict the ability to perform the functions in FMT_SMF.1 to authenticated administrators.

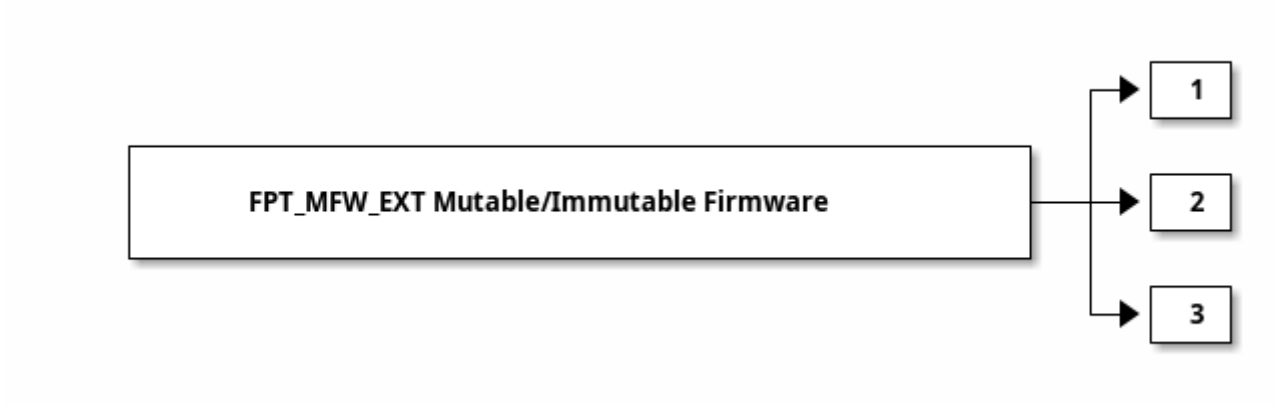
C.5. Class FPT: Protection of the TSF

C.5.1. FPT_MFW_EXT Mutable/Immutable Firmware

Family Behavior

This family defines requirements for specified types of firmware and the management of integrity and authenticity.

Component Leveling



FPT_MFW_EXT.1 Mutable/Immutable Firmware, requires the TSF to identify whether its firmware resides in mutable or immutable storage.

FPT_MFW_EXT.2 Basic Firmware Integrity, requires the TSF to assert the integrity of the firmware.

FPT_MFW_EXT.3 Firmware Authentication with Identity of Guarantor, requires the TSF to assert the authenticity of the firmware.

Management: FPT_MFW_EXT.1

The following actions could be considered for the management functions in FMT:

- Update TOE firmware and pre-installed SDOs.

Management: FPT_MFW_EXT.2, FPT_MFW_EXT.3

No specific management functions are identified.

Audit: FPT_MFW_EXT.1, FPT_MFW_EXT.2, FPT_MFW_EXT.3

There are no auditable events foreseen.

FPT_MFW_EXT.1 Mutable/Immutable Firmware

Hierarchical to	No other components.
Dependencies	No dependencies.

FPT_MFW_EXT.1.1

The TSF shall be maintained as [selection: *immutable, mutable*] firmware.

FPT_MFW_EXT.2 Basic Firmware Integrity

Hierarchical to	No other components.
Dependencies	FPT_MFW_EXT.1 Mutable/Immutable Firmware FCS_COP.1 Cryptographic Operation

FPT_MFW_EXT.2.1

The TSF shall have the ability to verify the integrity of the firmware.

FPT_MFW_EXT.2.2

The TSF shall provide a capability to generate evidence of the integrity of the firmware.

FPT_MFW_EXT.3 Firmware Authentication with Identity of Guarantor

Hierarchical to	No other components.
Dependencies	FPT_MFW_EXT.1 Mutable/Immutable Firmware FCS_COP.1 Cryptographic Operation

FPT_MFW_EXT.3.1

The TSF shall have the ability to verify the authenticity of the firmware.

FPT_MFW_EXT.3.2

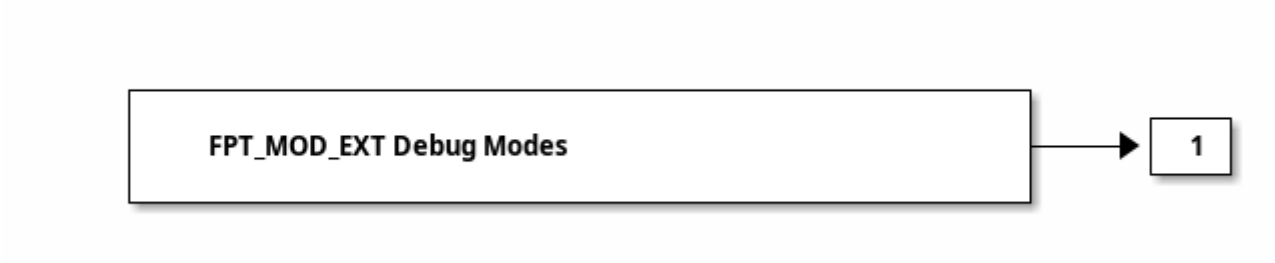
The TSF shall provide a capability to generate evidence of the authenticity of the firmware.

C.5.2. FPT_MOD_EXT Debug Modes

Family Behavior

This family defines requirements for debug modes.

Component Leveling



FPT_MOD_EXT.1 Debug Modes, requires the TSF to deny access to debug modes.

Management: FPT_MOD_EXT.1

No specific management functions are identified.

Audit: FPT_MOD_EXT.1

There are no auditable events foreseen.

FPT_MOD_EXT.1 Debug Modes

Hierarchical to	No other components.
Dependencies	No dependencies.

FPT_MOD_EXT.1.1

The TSF shall provide no access to debug modes.

C.5.3. FPT_PRO_EXT Root of Trust

Family Behavior

This family defines requirements for the TOE's implementation of a Root of Trust and its ability to use this to assert the integrity of its stored data.

Component Leveling

FPT_PRO_EXT.1 Root of Trust, requires the TSF to maintain a Root of Trust and identify how it is stored in memory.

FPT_PRO_EXT.2 Data Integrity Measurements, requires the TSF to generate integrity measurements to assert its own integrity.

Management: FPT_PRO_EXT.1, FPT_PRO_EXT.2

No specific management functions are identified.

Audit: FPT_PRO_EXT.1, FPT_PRO_EXT.2

There are no auditable events foreseen.

FPT_PRO_EXT.1 Root of Trust

Hierarchical to	No other components.
Dependencies	No dependencies.

FPT_PRO_EXT.1.1

The TSF shall contain an SDO that contains the identity of the Root of Trust.

FPT_PRO_EXT.1.2

The TSF shall maintain Root of Trust data as [selection: *immutable, mutable if and only if its mutability is controlled by a unique identifiable owner*].

FPT_PRO_EXT.2 Data Integrity Measurements

Hierarchical to	No other components.
Dependencies	No dependencies.

FPT_PRO_EXT.2.1

The TSF shall be able to quantify the integrity of the data protected by the TOE by generating integrity measurements and assertions.

FPT_PRO_EXT.2.2

The TSF shall accumulate platform characteristics using a consistent [assignment: *description of process for accumulating platform characteristics*] process in which verified quantifiable measurements and assertions are accumulated by the RoT for Measurement to prove the integrity of its SDOs.

C.5.4. FPT_ROT_EXT Root of Trust Services

Family Behavior

This family defines requirements for individual Root of Trust services that the TSF may implement.

Component Leveling

FPT_ROT_EXT.1 Root of Trust Services, requires the TSF to identify the specific Roots of Trust it provides.

FPT_ROT_EXT.2 Root of Trust for Storage, requires the TSF to prevent unauthorized access to SDOs associated with its Root of Trust for Storage.

FPT_ROT_EXT.3 Root of Trust for Reporting Mechanisms, requires the TSF to implement a Root of Trust for Reporting in the specified manner.

Management: FPT_ROT_EXT.1, FPT_ROT_EXT.2, FPT_ROT_EXT.3

No specific management functions are identified.

Audit: FPT_ROT_EXT.1, FPT_ROT_EXT.2, FPT_ROT_EXT.3

There are no auditable events foreseen.

FPT_ROT_EXT.1 Root of Trust Services

Hierarchical to	No other components.
Dependencies	FPT_PRO_EXT.1 Root of Trust

FPT_ROT_EXT.1.1

The TSF shall provide a Root of Trust for Storage, a Root of Trust for Authorization, and [selection: *Root of Trust for Measurement, Root of Trust for Reporting, no others*].

FPT_ROT_EXT.2 Root of Trust for Storage

Hierarchical to	No other components.
Dependencies	FPT_PRO_EXT.1 Root of Trust

FPT_ROT_EXT.2.1

The TSF shall prevent unauthorized access to SDOs associated with the Root of Trust for Storage.

FPT_ROT_EXT.3 Root of Trust for Reporting Mechanisms

Hierarchical to	No other components.
Dependencies	FCS_COP.1 Cryptographic Operation FPT_PRO_EXT.1 Root of Trust FPT_ROT_EXT.1 Root of Trust Services

FPT_ROT_EXT.3.1

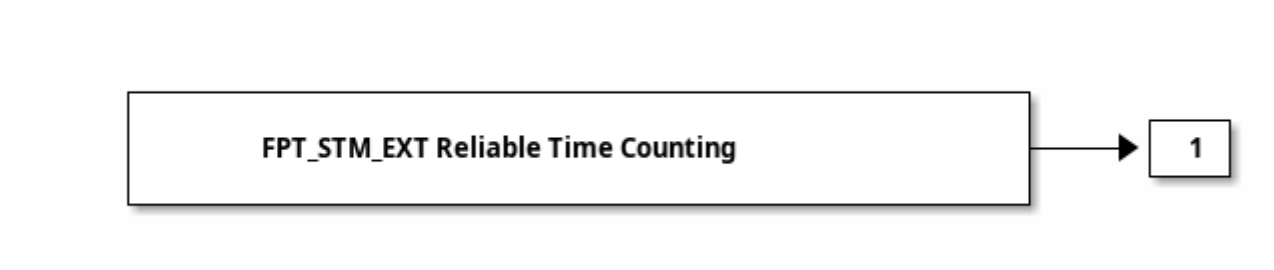
The TSF shall be able to attest to a state as represented by platform characteristics with a Root of Trust for Reporting mechanism that uses for its identity [selection: *a cryptographically verifiable identity in FPT_PRO_EXT.1, an alias key bound to the cryptographically verifiable identity in FPT_PRO_EXT.1*] and using a signature algorithm as specified in FCS_COP.1.

C.5.5. FPT_STM_EXT Reliable Time Counting

Family Behavior

This family defines requirements for time counting.

Component Leveling



FPT_STM_EXT.1 provides various methods for providing reliable time services.

Management: FPT_STM_EXT.1

No specific management functions are identified.

Audit: FPT_STM_EXT.1

There are no auditable events foreseen.

FPT_STM_EXT.1 Reliable Time Counting

Hierarchical to	No other components.
Dependencies	No dependencies.

FPT_STM_EXT.1 Reliable Time Counting

FPT_STM_EXT.1.1

The TSF shall be able to provide a reliable [selection: *internal time stamp, external time stamp, monotonically increasing counter*] to measure the passage of time.

C.6. Class FTP: Trusted Path/Channels

C.6.1. FTP_ITC_EXT Inter-TSF Trusted Channel

Family Behavior

This family defines requirements for the implementation of trusted communications with entities external to the TOE.

Component Leveling

FTP_ITC_EXT.1 Cryptographically Protected Communications Channels, requires the TSF to implement a cryptographic protocol as a method of communicating securely with external entities.

Management: FTP_ITC_EXT.1

No specific management functions are identified.

Audit: FTP_ITC_EXT.1

There are no auditable events foreseen.

FTP_ITC_EXT.1 Cryptographically Protected Communications Channels

Hierarchical to	No other components.
Dependencies	FCS_COP.1 Cryptographic Operation

FTP_ITC_EXT.1.1

The TSF shall use [assignment: *cryptographic protocol*] to provide a communication channel between itself and [assignment: *list of entities external to the TOE*] that protects channel data from disclosure and ensures the integrity of channel data.

C.6.2. FTP_ITE_EXT Encrypted Data Communications

Family Behavior

This family defines requirements for encryption of TSF data that is transmitted to an external entity over an insecure channel.

Component Leveling

FTP_ITE_EXT.1 Encrypted Data Communications, requires the TSF to encrypt data in the specified manner using key data that is provided to an external entity in the specified manner.

Management: FTP_ITE_EXT.1

No specific management functions are identified.

Audit: FTP_ITE_EXT.1

There are no auditable events foreseen.

FTP_ITE_EXT.1 Encrypted Data Communications

Hierarchical to	No other components.
Dependencies	FCS_COP.1 Cryptographic Operation

FTP_ITE_EXT.1.1

The TSF shall encrypt data for transfer between the TOE and [assignment: *list of entities external to the TOE*] using a cryptographic algorithm and key size as specified in FCS_COP.1, and using [assignment: *keys, identified by how they are generated by or imported into the TOE*].

C.6.3. FTP_ITP_EXT Physically Protected Channel

Family Behavior

This family defines requirements for use of physically protected communications mechanisms.

Component Leveling

FTP_ITP_EXT.1 Physically Protected Channel, requires the TSF to use a physically protected channel for transmission of data to an external entity.

Management: FTP_ITP_EXT.1

No specific management functions are identified.

Audit: FTP_ITP_EXT.1

There are no auditable events foreseen.

FTP_ITP_EXT.1 Physically Protected Channel

Hierarchical to No other components.

Dependencies No dependencies.

FTP_ITP_EXT.1.1

The TSF shall provide a physically protected communication channel between itself and [assignment: *list of other IT entities within the same platform*].

Appendix D: Entropy Documentation and Assessment

This appendix describes the required supplementary information for each entropy source used by the TOE.

The documentation of the entropy sources should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide sufficient entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS in the public facing ST.

D.1. Design Description

Documentation shall include the design of each entropy source as a whole, including the interaction of all entropy source components. Any information that can be shared regarding the design should also be included for any third-party entropy sources that are included in the product.

The documentation will describe the operation of the entropy source to include how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the entropy comes from, where the entropy output is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

If implemented, the design description shall include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on shall be included.

D.2. Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

The entropy justification shall not include any data added from any third-party application or from any state saving between restarts.

D.3. Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

D.4. Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, TOE behavior upon entropy source failure, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

Appendix E: SFR Dependencies Analysis

Table 25. SFR Dependencies Rationale for Mandatory SFRs

SFR	Dependencies	Rationale Statement
FCS_CKM.1	[FCS_CKM.1/AKG, FCS_CKM.1/SKG, FCS_CKM.2, FCS_CKM.5, FCS_CKM_EXT.7, FCS_CKM_EXT.8 or FCS_COP.1] [FCS_RBG.1 or FCS_RNG.1] FCS_CKM.6	FCS_CKM.1/AKG - (selection-based SFR) included FCS_CKM.1/SKG - (selection-based SFR) included FCS_CKM.2 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_COP.1 - included FCS_RBG.1 - included FCS_RNG.1 is satisfied by FCS_RBG.1 - included FCS_CKM.6 - included
FCS_CKM.2	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5 or FCS_COP.1]	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_COP.1/KeyEncap - (selection-based SFR) included FCS_COP.1/KeyWrap - (selection-based SFR) included FCS_COP.1/AEAD - (selection-based SFR) included

SFR	Dependencies	Rationale Statement
FCS_CKM.6	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1 or FCS_CKM.5]	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included
FCS_COP.1/Hash	No dependencies.	There are no keys involved with hashing, so no cryptograhpic key-based dependencies are necessary.
FCS_COP.1/KeyedHash	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5, FCS_CKM_EXT.7 or FCS_CKM_EXT.8] FCS_CKM.6	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_CKM.6 - included
FCS_COP.1/SigGen	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1/AKG, or FCS_CKM.5] FCS_CKM.6	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1/AKG - (selection-based SFR) included FCS_CKM.5 - (selection-based SFR) included FCS_CKM.6 - included

SFR	Dependencies	Rationale Statement
FCS_COP.1/SigVer	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, or FCS_CKM.5] FCS_CKM.6	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM.6 - included
FCS_COP.1/SKC	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5, FCS_CKM_EXT.7 or FCS_CKM_EXT.8] FCS_CKM.6 FCS_OTV_EXT.1	FDP_ITC_EXT.1 satisfies FDP_ITC.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC_EXT.2 satisfies FDP_ITC.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_CKM.6 - included FCS_OTV_EXT.1 - included
FCS_RBG.1	[FCS_RBG.2 or FCS_RBG.3] FPT_FLS.1 FPT_TST.1	FCS_RBG.2 - (optional SFR) included FCS_RBG.3 - (optional SFR) included FPT_FLS.1 - included FPT_TST.1 - included
FCS_OTV_EXT.1	FCS_RBG.1	FCS_RBG.1 - included
FCS_STG_EXT.1	No dependencies	N/A
FDP_ACC.1	FDP_ACF.1	FDP_ACF.1 - included

SFR	Dependencies	Rationale Statement
FDP_ACF.1	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1 - included FMT_MSA.3 - included
FDP_ETC_EXT.2	FCS_COP.1	FCS_COP.1 - included
FDP_FRS_EXT.1	FDP_FRS_EXT.2	FDP_FRS_EXT.2 - (selection-based SFR) included
FDP_ITC_EXT.1	FCS_COP.1 [FTP_ITC_EXT.1, FTP_ITE_EXT.1, or FTP_ITP_EXT.1]	FCS_COP.1 - included FTP_ITC_EXT.1 - (selection-based SFR) included FTP_ITE_EXT.1 - (selection-based SFR) included FTP_ITP_EXT.1 - (selection-based SFR) included
FDP_ITC_EXT.2	FCS_COP.1 [FTP_ITC_EXT.1, FTP_ITE_EXT.1, or FTP_ITP_EXT.1]	FCS_COP.1 - included FTP_ITC_EXT.1 - (selection-based SFR) included FTP_ITE_EXT.1 - (selection-based SFR) included FTP_ITP_EXT.1 - (selection-based SFR) included
FDP_RIP.1	No dependencies	N/A
FDP_SDC.2	FCS_COP.1	FCS_COP.1 - included
FDP_SDI.2	No dependencies	N/A
FIA_AFL_EXT.1	No dependencies	N/A
FIA_SOS.2	No dependencies	N/A
FIA_UAU.2	FIA_UID.1	This dependency is not present in the PP because all of the methods used to access the TSF (physically protected channels, encrypted data buffers, or cryptographically protected data channels) all implicitly identify the subject that is attempting to authenticate to the TOE.
FIA_UAU.5	No dependencies	N/A
FIA_UAU.6	No dependencies	N/A
FMT_MOF_EXT.1	FMT_SMF.1	FMT_SMF.1 - included
FMT_MSA.1	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	FDP_ACC.1 - included FMT_SMF.1 - included FMT_SMR.1 - included

SFR	Dependencies	Rationale Statement
FMT_MSA.3	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1 - included FMT_SMR.1 - included
FMT_SMF.1	No dependencies	N/A
FMT_SMR.1	FIA_UID.1	This dependency is not present in the PP because all of the methods used to access the TSF (physically protected channels, encrypted data buffers, or cryptographically protected data channels) all implicitly identify the subject that is attempting to authenticate to the TOE.
FPT_FLS.1/FI	No dependencies	N/A
FPT_MFW_EXT.1	No dependencies.	N/A
FPT_MOD_EXT.1	No dependencies	N/A
FPT_PHP.3	No dependencies	N/A
FPT_PRO_EXT.1	No dependencies	N/A
FPT_ROT_EXT.1	FPT_PRO_EXT.1	FPT_PRO_EXT.1 - included
FPT_ROT_EXT.2	FPT_PRO_EXT.1	FPT_PRO_EXT.1 - included
FPT_RPL.1/Authorization	No dependencies	N/A
FPT_TST.1	No dependencies	N/A
FRU_FLT.1	FPT_FLS.1	FPT_FLS.1/FI - included

Table 26. SFR Dependencies Rationale for Optional SFRs

SFR	Dependencies	Rationale Statement
FCS_RBG.2	FCS_RBG.1	FCS_RBG.1 - included
FCS_RBG.3	FCS_RBG.1	FCS_RBG.1 - included
FCS_RBG.4	FCS_RBG.1 FCS_RBG.5	FCS_RBG.1 - included FCS_RBG.5 - (optional SFR) included
FCS_RBG.5	FCS_RBG.1 [FCS_RBG.2 or FCS_RBG.3 or RCS_RBG.4]	FCS_RBG.1 - included FCS_RBG.2 - (optional SFR) included FCS_RBG.3 - (optional SFR) included FCS_RBG.4 - (optional SFR) included
FCS_RBG.6	FCS_RBG.1	FCS_RBG.1 - included

SFR	Dependencies	Rationale Statement
FPT_ITT.1	No dependencies	N/A
FPT_PRO_EXT.2	No dependencies	N/A
FPT_ROT_EXT.3	FCS_COP.1 FPT_PRO_EXT.1 FPT_ROT_EXT.1	FCS_COP.1- included FPT_PRO_EXT.1 - included FPT_ROT_EXT.1 - included

Table 27. SFR Dependencies Rationale for Selection-Based SFRs

SFR	Dependencies	Rationale Statement
FCS_CKM.1/AKG	[FCS_CKM.2, FCS_CKM.5 or FCS_COP.1] [FCS_RBG.1 or FCS_RNG.1] FCS_CKM.6	FCS_CKM.2 - included FCS_CKM.5 - (selection-based SFR) included FCS_COP.1 - included FCS_RBG.1 - included FCS_RNG.1 is satisfied by FCS_RBG.1 - included FCS_CKM.6 - included
FCS_CKM.1/SKG	[FCS_CKM.2, FCS_CKM.5, FCS_CKM_EXT.7 or FCS_COP.1] [FCS_RBG.1 or FCS_RNG.1] FCS_CKM.6	FCS_CKM.2 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_COP.1 - included FCS_RBG.1 - included FCS_RNG.1 is satisfied by FCS_RBG.1 - included FCS_CKM.6 - included

SFR	Dependencies	Rationale Statement
FCS_CKM_EXT.3	[FCS_CKM.1, FCS_CKM.5 or FCS_CKM_EXT.8] [FCS_COP.1/KeyEncap, FCS_COP.1/KeyWrap, FCS_COP.1/SKC or FCS_COP.1/AEAD] FCS_CKM.6	FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_COP.1/KeyEncap - (selection-based SFR) included FCS_COP.1/KeyWrap - (selection-based SFR) included FCS_COP.1/SKC - included FCS_COP.1/AEAD - (selection-based SFR) included FCS_CKM.6 - included
FCS_CKM.5	FCS_CKM.2 FCS_COP.1 FCS_CKM.6	FCS_CKM.2 - included FCS_COP.1 - included FCS_CKM.6 - included
FCS_CKM_EXT.7	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5, FCS_CKM_EXT.8] [FCS_CKM.2 or FCS_COP.1] FCS_CKM.6	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_CKM.2 - included FCS_COP.1 - included FCS_CKM.6 - included

SFR	Dependencies	Rationale Statement
FCS_CKM_EXT.8	[FCS_CKM.2 or FCS_CKM_EXT.7] FCS_COP.1/KeyedHash FCS_CKM.6 FCS_OTV_EXT.1	FCS_CKM.2 - included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_COP.1/KeyedHash - included FCS_CKM.6 - included FCS_OTV_EXT.1 - included
FCS_COP.1/AEAD	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5, FCS_CKM_EXT.7 or FCS_CKM_EXT.8] FCS_CKM.6 FCS_OTV_EXT.1	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_CKM.6 - included FCS_OTV_EXT.1 - included
FCS_COP.1/CMAC	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5, FCS_CKM_EXT.7 or FCS_CKM_EXT.8] FCS_CKM.6	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_CKM.6 - included

SFR	Dependencies	Rationale Statement
FCS_COP.1/Key Encap	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5, FCS_CKM_EXT.7, or FCS_CKM_EXT.8] FCS_CKM.6 FCS_OTV_EXT.1	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_CKM.6 - included FCS_OTV_EXT.1 - included
FCS_COP.1/Key Wrap	[FDP_ITC.1, FDP_ITC.2, FCS_CKM.1, FCS_CKM.5, FCS_CKM_EXT.7, or FCS_CKM_EXT.8] FCS_CKM.6	FDP_ITC.1 is satisfied by FDP_ITC_EXT.1 related to a mechanism by which key data can be imported into the TSF - included FDP_ITC.2 is satisfied by FDP_ITC_EXT.2 related to a mechanism by which key data can be imported into the TSF - included FCS_CKM.1 - included FCS_CKM.5 - (selection-based SFR) included FCS_CKM_EXT.7 - (selection-based SFR) included FCS_CKM_EXT.8 - (selection-based SFR) included FCS_CKM.6 - included
FDP_DAU.1/Prove	No dependencies	N/A
FDP_FRS_EXT.2	FDP_FRS_EXT.1	FDP_FRS_EXT.1 - included
FIA_AFL_EXT.2	FIA_AFL_EXT.1	FIA_AFL_EXT.1 - included
FPT_FLS.1/FW	No dependencies	N/A

SFR	Dependencies	Rationale Statement
FPT_MFW_EXT.2	FPT_MFW_EXT.1 FCS_COP.1	FPT_MFW_EXT.1 - included FCS_COP.1 - included
FPT_MFW_EXT.3	FPT_MFW_EXT.1 FCS_COP.1	FPT_MFW_EXT.1 - included FCS_COP.1 - included
FPT_RPL.1/Rollback	No dependencies	N/A
FPT_STM_EXT.1	No dependencies	N/A
FTP_ITC_EXT.1	FCS_COP.1	FCS_COP.1 - included
FTP_ITE_EXT.1	FCS_COP.1	FCS_COP.1 - included
FTP_ITP_EXT.1	No dependencies	N/A

Appendix F: SFR Architecture

A DSC implements all seven services in Table 28 as well as self-protection functionality that protects against a compromise or degradation of these services.

Table 28. SFR Architecture

Service	Applicable Requirements	
Parse	FCS_CKM.1	Cryptographic Key Generation
	FCS_CKM_EXT.7	Cryptographic Key Agreement
	FCS_CKM_EXT.8	Password-Based Key Derivation
	FCS_COP.1/AEAD	Cryptographic Operation (Authenticated Encryption with Associated Data)
	FCS_COP.1/Hash	Cryptographic Operation - Hashing
	FCS_COP.1/KeyedHash	Cryptographic Operation - Keyed Hash
	FCS_COP.1/CMAC	Cryptographic Operation - CMAC
	FCS_COP.1/KeyEncap	Cryptographic Operation - Key Encapsulation
	FCS_COP.1/KeyWrap	Cryptographic Operation - Key Wrapping
	FCS_COP.1/SKC	Cryptographic Operation - Symmetric-Key Cryptography
	FDP_ACC.1	Subset Access Control
	FDP_ACF.1	Security Attribute Based Access Control
	FDP_ITC_EXT.1	Parsing of SDEs
	FDP_ITC_EXT.2	Parsing of SDOs
	FTP_ITP_EXT.1	Physically Protected Channel
	FTP_ITC_EXT.1	Cryptographically Protected Communications Channels
	FTP_ITE_EXT.1	Encrypted Data Communications

Service	Applicable Requirements	
Provision	FCS_CKM.1/AKG	Cryptographic Key Generation - Asymmetric Key
	FCS_CKM.5	Cryptographic Key Derivation
	FCS_COP.1/AEAD	Cryptographic Operation (Authenticated Encryption with Associated Data)
	FCS_COP.1/Hash	Cryptographic Operation - Hashing
	FCS_COP.1/KeyedHash	Cryptographic Operation - Keyed Hash
	FCS_COP.1/CMAC	Cryptographic Operation - CMAC
	FCS_COP.1/SKC	Cryptographic Operation - Symmetric-Key Cryptography
	FCS_RBG.1	Random Bit Generation (RBG)
	FDP_ACC.1	Subset Access Control
	FDP_ACF.1	Security Attribute Based Access Control
	FIA_SOS.2	TSF Generation of Secrets
	FMT_MSA.3	Static Attribute Initialization
	FPT_STM_EXT.1	Reliable Time Counting
	FCS_RBG.6	Random Bit Generation Service
	FCS_RBG.2	Random Bit Generation (External Seeding)
	FCS_RBG.3	Random Bit Generation (Internal Seeding - Single Source)
	FCS_RBG.4	FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)
	FCS_RBG.5	Random Bit Generation (Combining Entropy Sources)
	FCS_CKM.1/SKG	Cryptographic Key Generation - Symmetric Key

Service	Applicable Requirements	
Protect	FCS_COP.1/AEAD	Cryptographic Operation (Authenticated Encryption with Associated Data)
	FCS_COP.1/Hash	Cryptographic Operation - Hashing
	FCS_COP.1/KeyedHash	Cryptographic Operation - Keyed Hash
	FCS_COP.1/CMAC	Cryptographic Operation - CMAC
	FCS_COP.1/SKC	Cryptographic Operation - Symmetric-Key Cryptography
	FCS_STG_EXT.1	Protected Storage
	FDP_SDC.2	Stored data confidentiality with dedicated method
	FDP_SDI.2	Stored Data Integrity Monitoring and Action
	FMT_SMR.1	Separation of Roles
	FPT_FLS.1/FI	Failure with Preservation of Secure State (Fault Injection)
	FPT_MOD_EXT.1	Debug Modes
	FPT_PHP.3	Resistance to Physical Attack
	FPT_ROT_EXT.1	Root of Trust Services
	FPT_ROT_EXT.2	Root of Trust for Storage
	FPT_PRO_EXT.2	Data Integrity Measurements
	FDP_FRS_EXT.2	Factory Reset Behavior
	FIA_AFL_EXT.2	Authorization Failure Response
	FPT_FLS.1/FW	Failure with Preservation of Secure State (Firmware)
	FPT_ITT.1	Basic Internal TSF Data Transfer Protection

Service	Applicable Requirements	
Process	FCS_COP.1/AEAD	Cryptographic Operation (Authenticated Encryption with Associated Data)
	FCS_COP.1/Hash	Cryptographic Operation - Hashing
	FCS_COP.1/KeyedHash	Cryptographic Operation - Keyed Hash
	FCS_COP.1/CMAC	Cryptographic Operation - CMAC
	FCS_COP.1/KeyEnc	Cryptographic Operation (Key Encryption)
	FCS_COP.1/SigGen	Cryptographic Operation - Signature Generation
	FCS_COP.1/SigVer	Cryptographic Operation - Signature Verification
	FCS_COP.1/SKC	Cryptographic Operation - Symmetric-Key Cryptography
	FCS_OTV_EXT.1	One-Time Value
	FDP_ACC.1	Subset Access Control
	FDP_ACF.1	Security Attribute Based Access Control
	FIA_AFL_EXT.1	Authorization Failure Handling
	FIA_SOS.2	TSF Generation of Secrets
	FIA_UAU.2	User Authentication before any Action
	FIA_UAU.5	Multiple Authentication Mechanisms
	FIA_UAU.6	Re-Authenticating
	FMT_MOF_EXT.1	Management of Security Functions Behavior
	FMT_MSA.1	Management of Security Attributes
	FMT_SMF.1	Specification of Management Functions
	FMT_SMR.1	Separation of Roles
	FPT_ROT_EXT.1	Root of Trust Services
	FPT_RPL.1/Authorization	Replay Prevention
	FPT_STM.1	Reliable Time Counting
	FIA_AFL_EXT.2	Authorization Failure Response

Service	Applicable Requirements	
Prove	FCS_COP.1/Hash	Cryptographic Operation - Hashing
	FCS_COP.1/KeyedHash	Cryptographic Operation - Keyed Hash
	FCS_COP.1/CMAC	Cryptographic Operation - CMAC
	FCS_RBG.1	Random Bit Generation (RBG)
	FCS_OTV_EXT.1	One-Time Value
	FDP_ACC.1	Subset Access Control
	FDP_ACF.1	Security Attribute Based Access Control
	FPT_PRO_EXT.1	Root of Trust
	FPT_RPL.1/Authorization	Replay Prevention
	FPT_STM_EXT.1	Reliable Time Counting
	FCS_RBG.2	Random Bit Generation (External Seeding)
	FCS_RBG.3	Random Bit Generation (Internal Seeding - Single Source)
	FCS_RBG.4	FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)
	FCS_RBG.5	Random Bit Generation (Combining Entropy Sources)
	FPT_ROT_EXT.3	Root of Trust for Reporting Mechanisms
	FDP_DAU.1/Prove	Basic Data Authentication (for Use with the Prove Service)
	FPT_MFW_EXT.1	Mutable/Immutable Firmware
	FPT_MFW_EXT.2	Basic Firmware Integrity
	FPT_MFW_EXT.3	Firmware Authentication with Identity of Guarantor

Service	Applicable Requirements	
Propagate	FCS_COP.1/AEAD	Cryptographic Operation (Authenticated Encryption with Associated Data)
	FCS_COP.1/Hash	Cryptographic Operation - Hashing
	FCS_COP.1/KeyedHash	Cryptographic Operation - Keyed Hash
	FCS_COP.1/CMAC	Cryptographic Operation - CMAC
	FCS_COP.1/KeyEnc	Cryptographic Operation (Key Encryption)
	FCS_COP.1/SKC	Cryptographic Operation - Symmetric-Key Cryptography
	FCS_RBG.1	Random Bit Generation (RBG)
	FCS_OTV_EXT.1	One-Time Value
	FDP_ACC.1	Subset Access Control
	FDP_ACF.1	Security Attribute Based Access Control
	FDP_ETC_EXT.2	Propagation of SDOs
	FCS_RBG.2	Random Bit Generation (External Seeding)
	FCS_RBG.3	Random Bit Generation (Internal Seeding - Single Source)
	FCS_RBG.4	FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)
	FCS_RBG.5	Random Bit Generation (Combining Entropy Sources)
	FPT_ITT.1	Basic Internal TSF Data Transfer Protection
	FTP_ITP_EXT.1	Physically Protected Channel
	FTP_ITC_EXT.1	Cryptographically Protected Communications Channels
	FTP_ITE_EXT.1	Encrypted Data Communications
Purge	FCS_CKM.6	Timing and event of cryptographic key destruction
	FCS_RBG.1	Random Bit Generation (RBG)
	FDP_RIP.1	Subset Residual Information Protection
	FCS_RBG.2	Random Bit Generation (External Seeding)
	FCS_RBG.3	Random Bit Generation (Internal Seeding - Single Source)
	FCS_RBG.4	FCS_RBG.4 Random Bit Generation (Internal Seeding - Multiple Sources)
	FCS_RBG.5	Random Bit Generation (Combining Entropy Sources)
	FDP_FRS_EXT.2	Factory Reset Behavior

Service	Applicable Requirements	
TSF Security	FDP_FRS_EXT.1	Factory Reset
	FPT_MFW_EXT.1	Mutable/Immutable Firmware
	FMT_SMF.1	Specification of Management Functions
	FPT_FLS.1/FI	Failure with Preservation of Secure State (Fault Injection)
	FPT_MOD_EXT.1	Debug Modes
	FPT_PHP.3	Resistance to Physical Attack
	FPT_TST.1	TSF Testing
	FRU_FLT.1	Degraded Fault Tolerance
	FPT_PRO_EXT.2	Data Integrity Measurements
	FPT_MFW_EXT.2	Basic Firmware Integrity
	FPT_MFW_EXT.3	Firmware Authentication with Identity of Guarantor
	FDP_FRS_EXT.2	Factory Reset Behavior
	FPT_FLS.1/FW	Failure with Preservation of Secure State (Firmware)
	FPT_RPL.1/Rollback	Replay Detection (Rollback)

Appendix G: Glossary

Table 29. Glossary

Term	Meaning
Access	In the context of SDOs, access to an SDO represents the list of actions permissible with an SDO, including its generation, use, modification, propagation, and destruction.
Administrator	A type of user that has special privileges to manage the TSF.
Attestation	The process of presenting verifiable evidence describing those characteristics that affect integrity. Examples of these characteristics are boot firmware and boot critical data which, combined, describe the way the DSC booted. [SA]
Attributes	Indications of characteristics or properties of the SDEs bound in an SDO.
Authorization Value	Critical data bound to an action by itself or to action on a subject. Such data, when presented to the TOE, authorizes the action by itself or authorizes the action on or with the subject respectively.
Authorization Data	Collective term for authentication tokens and authorization values.
Authentication Token	Critical data bound to a user. Such data, when presented to the TOE and successfully verified by it, authenticates the user. The TOE may use the successful authentication of a user as an authorization to execute an action on its behalf.
Authenticator	A shortened name for Authentication Token.
Boot Critical Data	Critical data that persists across power cycles and determines characteristics of the DSC. Examples of boot critical data can be DSC configuration settings, certificates, and the results of measurements obtained by the RoT for measurement.
Boot Firmware	The first firmware that executes during the boot process.
Chain of Trust	A Chain of Trust is anchored in a RoT and extends a trust boundary by verifying the authenticity and integrity of successive components before passing control to those components. [SA]
Client Application	Entity who relies on the services provided by the platform or DSC.
Data Encryption Key	An encryption key, usually for a symmetric algorithm, that encrypts data that is not keying material.
Integrity	Assurance of trustworthiness and accuracy.
Immutable	Unchangeable.
Key Encryption Key	An encryption key that encrypts other keying material. This is sometimes called a key wrapping key. A KEK can be either symmetric or asymmetric.
Known Answer Tests (KATs)	Test vectors or data generated to determine the correctness of an implementation.

Term	Meaning
Operator	Human being who has physical possession of the platform on which the DSC is located. [GD]
Owner	Human being who controls/manages the platform on which the DSC is located. May be remote. [GD]
Permanent Keys/Seeds	Keys or seeds that are provisioned to the device during manufacturing or initial setup that remain even after a factory reset.
Persistent Secrets	Persistent secrets are long term keys or key material that are maintained longer than a single cryptographic operation. Persistent secrets do not remain after a factory reset.
Platform	A platform consists of the hardware and firmware of a computing entity.
Pre-installed SDO	An SDO installed on the DSC by the manufacturer. The SDO consists of an SDE and attributes, which if not explicitly expressed in a data structure, are implicit based on the functions that have exclusive access to the SDE.
Privileged Function	Functions restricted to the ADM-R role, which may include, but are not limited to, provisioning keys, provisioning user authorization values, de-provisioning user authorization values, provisioning administrator authorization values, changing authorization values, disabling key escrow, and configuring cryptography.
Protected Data Blob	Data in an encrypted structure that protects its confidentiality or integrity (as required by the context in which it is used).
Protected Storage	Protected Storage usually refers to DSC hardware used to store SDEs or SDOs, and provide integrity protection for all items and confidentiality for those items that require it. Protected Storage may also refer to storage external to the DSC, which is usually encrypted by keys maintained by the DSC's internal protected storage capabilities.
Protections	Mechanisms that ensure components of a DSC (executable firmware code and critical data) remain in a state of integrity and are protected from modification outside of authorized, authenticated processes and entities. [NIST-ROTM]
Remote Secure Channel	Logical channel to the DSC from a remote entity, which cryptographically protects the confidentiality and integrity of the channel content.
Root Encryption Key	An encryption key that serves as the anchor of a hierarchy of keys.
Root of Trust (RoT)	A RoT performs one or more security specific functions; establishing the foundation on which all trust in a system is placed. [NIST-ROTM]
RoT for Authorization	(As defined by [GP_ROT]) The RoT for Authorization provides reliable capabilities to assess authorization tokens and determine whether or not they satisfy policies for access control.
RoT for Confidentiality	(As defined by [GP_ROT]) The RoT for Confidentiality maintains shielded locations for the purpose of storing sensitive data, such as secret keys and passwords.

Term	Meaning
RoT for Integrity	(As defined by [GP_ROT]) The RoT for Integrity maintains shielded locations for the purpose of storing and protecting the integrity of non-secret critical security parameters and platform characteristics. Critical security parameters include, but are not limited to, authorization values, public keys, and public key certificates.
RoT for Measurement	(As defined by [GP_ROT]) The RoT for Measurement provides the ability to reliably create platform characteristics.
RoT for Reporting	(As defined by [GP_ROT]) The RoT for Reporting reliably reports platform characteristics. It provides an interface that limits its services to providing reports on its platform characteristics authenticated by a platform identity.
RoT for Storage	A RoT that acts as the RoT for Confidentiality and the RoT for Integrity.
RoT for Update	A RoT responsible for updating the firmware.
RoT for Verification	A RoT responsible for verifying digital signatures.
Security Data Element (SDE)	A Critical Security Parameter, such as a cryptographic key or authorization token.
Security Data Object (SDO)	An SDO may include one or more SDEs. SDOs bind SDEs with a set of attributes.
Symmetric Encryption Key	A value intend to input as a key to a symmetric encryption algorithm, such as AES.
System	A system consists of the platform hardware and firmware in addition to the higher-level software running on top of it (kernel, user-space processes, etc.).
Trusted Local Channel	Physical channel to the DSC within the platform of which the DSC is a part, which is protected by the operational environment to ensure confidentiality and integrity.
User	An administrator or client application.

See [CC1] for other Common Criteria abbreviations and terminology.

Appendix H: Acronyms

Table 30. Acronyms

Acronym	Meaning
AES	Advanced Encryption Standard
CApp	Client Application
CBC	Cipher Block Chaining
CCM	Counter with CBC-Message Authentication Code
CPU	Central Processing Unit
CSP	Critical Security Parameter
DAR	Data-At-Rest
DEK	Data Encryption Key
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIPS	Federal Information Processing Standards
FQDN	Fully Qualified Domain Name
GCM	Galois Counter Mode
HMAC	Keyed-Hash Message Authentication Code
HTTPS	Hypertext Transfer Protocol Secure
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ITSEF	Information Technology Security Evaluation Facility
KEK	Key Encryption Key
KMAC	KECCACK Message Authentication Code
NIST	National Institute of Standards and Technology
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PP	Protection Profile
RA	Registration Authority
RBG	Random Bit Generator
REK	Root Encryption Key
ROM	Read-only memory

Acronym	Meaning
RSA	Rivest Shamir Adleman Algorithm
SDE	Security Data Element
SDO	Security Data Object
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SK	Symmetric Key or Symmetric Encryption Key
SPI	Security Parameter Index
SSH	Secure Shell
ST	Security Target
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
USB	Universal Serial Bus

Appendix I: References & Standards

I.1. Standards

[AIS31] Matthias Peter and Werner Schindler. *A proposal for: Functionality classes for random number generators*, Bundesamt für Sicherheit in der Informationstechnik (BSI), September 2011

[ANSI-ECC] American National Standards Institute. *ANSI X9.63-2011 (R2017) Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography*, American National Standards Institute, February 2017

[FIPS-AES] National Institute of Standards and Technology. *Federal Information Processing Standard Publication (FIPS-PUB) 197, Advanced Encryption Standard (AES)*, National Institute of Standards and Technology, November 2001

[FIPS-DSS] National Institute of Standards and Technology. *Federal Information Processing Standard Publication (FIPS-PUB) 186-5, Digital Signature Standard (DSS)*, National Institute of Standards and Technology, February 2023

[FIPS-HMAC] National Institute of Standards and Technology. *Federal Information Processing Standard Publication (FIPS-PUB) 198-1, The Keyed-Hash Message Authentication Code (HMAC)*, National Institute of Standards and Technology, July 2008

[FIPS-SHA3] National Institute of Standards and Technology. *Federal Information Processing Standard Publication (FIPS-PUB) 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, National Institute of Standards and Technology, August 2015

[FIPS-SHS] National Institute of Standards and Technology. *Federal Information Processing Standard Publication (FIPS-PUB) 180-4, Secure Hash Standard (SHS)*, National Institute of Standards and Technology, August 2015

[IEEE-PK] Institute of Electrical and Electronics Engineers. *IEEE 1363a-2004 - IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques*, Institute of Electrical and Electronics Engineers, September 2004

[IEEE-XTS] Institute of Electrical and Electronics Engineers. *IEEE 1619-2018 - IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices*, Institute of Electrical and Electronics Engineers, January 2019

[ISO-ACIPH] ISO/IEC. *ISO/IEC 18033-2:2006 Information Technology - Security Techniques - Encryption Algorithms - Part 3: Asymmetric Ciphers*, ISO/IEC, May 2006

[ISO-AENC] ISO/IEC. *ISO/IEC 19772:2020 Information Technology - Authenticated Encryption*, ISO/IEC, November 2020

[ISO-BCIPH] ISO/IEC. *ISO/IEC 18033-3:2010 Information Technology - Security Techniques - Encryption Algorithms - Part 3: Block Ciphers*, ISO/IEC, December 2012

[ISO-CMAC] ISO/IEC. *ISO/IEC 9797-1:2011 Information Technology - Security Techniques - Message*

Authentication Codes (MACs) - Part 1: Mechanisms Using a Block Cipher, ISO/IEC, March 2011

[ISO-DSIG] ISO/IEC. *ISO/IEC 14888-3:2018 Information Technology - Security Techniques - Digital Signatures with Appendix - Part 3: Discrete Logarithm Based Mechanisms*, ISO/IEC, November 2018

[ISO-HASH] ISO/IEC. *ISO/IEC 10118-3:2018 Information Technology - Security Techniques - Hash-Functions - Part 3: Dedicated Hash-Functions*, ISO/IEC, October 2018

[ISO-KDF] ISO/IEC. *ISO/IEC 11770-6:2016 Information Technology - Security Techniques - Key Management - Part 6: Key Derivation*, ISO/IEC, October 2016

[ISO-LCBC] ISO/IEC. *ISO/IEC 29192-2:2019 Information Technology - Security Techniques - Lightweight Cryptography - Part 2: Block Ciphers*, ISO/IEC, December 2012

[ISO-MAC] ISO/IEC. *ISO/IEC 9797-2:2021 Information Technology - Security Techniques - Message Authentication Codes (MACs) - Part 2: Mechanisms Using a Dedicated Hash-Function*, ISO/IEC, June 2021

[ISO-MODES] ISO/IEC. *ISO/IEC 10116:2017 Information Technology - Security Techniques - Modes of Operation for an n-bit Block Cipher*, ISO/IEC, July 2017

[ISO-RBG] SO/IEC. *ISO/IEC 18031:2011 Information Technology - Security Techniques - Random Bit Generation*, ISO/IEC, November 2011

[ISO-TR] ISO/IEC. *ISO/IEC 24759-3:2017 Information Technology - Security Techniques - Test Requirements for Cryptographic Modules*, ISO/IEC, 2017

[NIST-ASKDF] Quynh Dang. *NIST Special Publication 800-135 Rev. 1, Recommendation for Existing Application-Specific Key Derivation Functions*, National Institute of Standards and Technology, December 2011

[NIST-CCM] Morris Dworkin. *NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, National Institute of Standards and Technology, May 2004

[NIST-CKG] Elaine Barker, Allen Roginsky, and Richard Davis. *NIST Special Publication 800-133 Rev. 2, Recommendation for Cryptographic Key Generation*, National Institute of Standards and Technology, June 2020

[NIST-CMAC] Morris Dworkin. *NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, National Institute of Standards and Technology, October 2016

[NIST-CURVES] Lily Chen, Dustin Moody, Karen Randall, Andrew Regenscheid, Angela Robinson, and Karen Randall. *NIST Special Publication 800-186, Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*, National Institute of Standards and Technology, February 2023

[NIST-DRBG] Elaine Barker and John Kelsey. *NIST Special Publication 800-90A Rev. 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, National Institute of Standards and Technology, June 2015

[NIST-ES] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A. McKay, Mary L. Baish, and Mike Boyle. *NIST Special Publication 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation*, National Institute of Standards and Technology, January 2018

[NIST-GCM] Morris Dworkin. *NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, National Institute of Standards and Technology, November 2007

[NIST-HBS] David Cooper, Daniel Apon, Quynh Dang, Michael Davidson, Morris Dworkin, and Carl Miller. *NIST Special Publication 800-208, Recommendation for Stateful Hash-Based Signature Schemes*, National Institute of Standards and Technology, October 2020

[NIST-KBKDF] Lily Chen. *NIST Special Publication 800-108 Rev. 1, Recommendation for Key Derivation Using Pseudorandom Functions*, National Institute of Standards and Technology, August 2022

[NIST-KDLC] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, and Richard Davis. *NIST Special Publication 800-56A Rev. 3, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*, National Institute of Standards and Technology, April 2018

[NIST-KDRV] Elaine Barker, Lily Chen, and Richard Davis. *NIST Special Publication 800-56C Rev. 2, Recommendation for Key-Derivation Methods in Key-Establishment Schemes*, National Institute of Standards and Technology, August 2020

[NIST-KDV] John Kelsey, Shu-jen Chang, and Ray Perlner. *NIST Special Publication 800-185, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash*, National Institute of Standards and Technology, December 2016

[NIST-KIFC] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, and Scott Simon. *NIST Special Publication 800-56B Rev. 2, Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography*, National Institute of Standards and Technology, March 2019

[NIST-KL] Elaine Barker and Allen Roginsky. *NIST Special Publication 800-131A Rev. 2, Transitioning the Use of Cryptographic Algorithms and Key Lengths*, National Institute of Standards and Technology, December 2019

[NIST-KW] Morris Dworkin. *NIST Special Publication 800-38F, Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*, National Institute of Standards and Technology, December 2012

[NIST-MODES] Morris Dworkin. *NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, National Institute of Standards and Technology, December 2001

[NIST-PBKDF] Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen. *NIST Special Publication 800-132, Recommendation for Password-Based Key Derivation, Part 1: Storage Applications*, National Institute of Standards and Technology, December 2010

[NIST-XTS] Morris Dworking. *NIST Special Publication 800-38E, Recommendation for Block Cipher*

Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices, National Institute of Standards and Technology, January 2010

[RFC3526] Mika Kojo and Tero Kivinen. *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*, Internet Engineering Task Force, May 2003

[RFC5639] Manfred Lochter and Johannes Merkle. *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*, Internet Engineering Task Force, March 2010

[RFC7748] Adam Langley, Mike Hamburg, and Sean Turner. *Elliptic Curves for Security*, Internet Engineering Task Force, January 2016

[RFC7919] Daniel Kahn Gillmor. *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)*, Internet Engineering Task Force, August 2016

[RFC8017] Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch. *PKCS #1: RSA Cryptography Specifications Version 2.2*, Internet Engineering Task Force, November 2016

[RFC8032] Simon Josefsson and Ilari Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*, Internet Engineering Task Force, January 2017

[RFC8391] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. *XMSS: eXtended Merkle Signature Scheme*, Internet Engineering Task Force, May 2018

[RFC8554] David McGrew, Michael Curcio, and Scott Fluhrer. *Leighton-Micali Hash-Based Signatures*, Internet Engineering Task Force, April 2019

[SECG-ECC] Daniel R. L. Brown. *SEC 1: Elliptic Curve Cryptography*, Certicom Corp., May 2009

I.2. References

[GD] Grawrock, David. *Dynamics of a Trusted Platform: A building block approach*. Intel Press, 2009

[GP_ROT] GlobalPlatform Technology. *Root of Trust Definitions and Requirements Version 1.1*. GlobalPlatform, June 2018

[NIST-ROTM] Lily Chen, Joshua Franklin, and Andrew Regenscheid. *NIST Special Publication 800-164 (Draft), Guidelines on Hardware Rooted Security in Mobile Devices (Draft)*, National Institute of Standards and Technology, October 2012

[SA] Segall, Ariel. *Trusted Platform Modules: Why, When and How to Use Them*. The Institution of Engineering and Technology, 2016