SECTON/CRATON2
Embedded V2X HSM CUT3.1
Security Target Lite
V2.0

Date: 2023/11/12

Created by

Change History

Version	Date	Author	Comment
0.1	2023/09/27	Autotalks Ltd.	Initial release
0.2	2023/09/29	Autotalks Ltd.	Sanitized
2.0	2023/11/12	Autotalks Ltd.	Update references in section “References” and in Table 1

Table of contents

1 ST Introduction

1.1 ST Reference

Title: SECTON/CRATON2 Embedded V2X HSM CUT3.1 Security Target Lite

Version: v2.0

Author: Autotalks Ltd.

Date of publication: 2023/11/12

1.2 TOE Reference

TOE Name: SECTON/CRATON2 Embedded V2X HSM CUT3.1

TOE Developer: Autotalks Ltd.

TOE Version: CUT 3.1

- Firmware version: 0xC1010400
- Hardware version: Wafer VA83D

1.3 TOE Overview

1.3.1 Introduction

The TOE (also called eHSM or V2X eHSM) is an HSM composed by hardware and software which is embedded in a SoC. This SoC can be delivered in two architectures, called SECTON or CRATON2, that constitute the operational environment of the TOE while the TOE contained in them, is the same.

The TOE is designed to be used in vehicle or in stationary deployments, for supporting global V2X (Vehicle to anything) communication devices for Cooperative Intelligent Transport System (C-ITS). The TOE provides secure cryptographic services and key management services to the VCS element that reside in the same SoC. The term used in this document for describing the SECTON/CRATON2 elements outside the TOE (operational environment) is the SoC Application Side (SaS).

Guidance documentation for the integration and operation of the TOE in its intended environment is also included.

1.3.2 TOE Type

The TOE is a Hardware Security Module composed by hardware and firmware which is embedded in a SoC. It provides cryptographic services to a communication device (VCS) in a Cooperative Intelligent Transport System (C-ITS).

The TOE is intended to be used in a vehicle or in stationary deployments. The TOE described in this document is a tamper-resistant hardware module including the firmware required for its functionality.

1.3.3 TOE Usage & Major Security Features

The TOE supports the VCS with cryptographic operations and key management functionality as specified in **[IEEE 1609.2]** and **[IEEE 1609.2.1]** supporting relevant ETSI ITS standards.

The TOE major security features are:

- V2X Key Management
- Digital signature generation
- User data ECIES encryption/decryption
- ECC Key derivation
- Random number generation
- Self-protection

1.3.3.1 V2X Key Management

The TOE handles generation, secure import, secure export and destruction of ECC private keys and symmetric keys.

The TOE generates ECC key pairs, which are used in:

- Digital signature following ECDSA algorithm
- Encryption and decryption following ECIES encryption scheme (refer to **[IEEE 1609.2]**).
- Key derivation as part of the Butterfly Key Expansion Mechanism (refer to **[IEEE 1609.2.1]**).

Generated keys are encapsulated inside a cryptographically protected blob structure to be exported outside the TOE.

In the V2X context, the following set of ECDSA keys in the context of V2X communications will be generated:

- Canonical Key: used to sign initial EC request;
- Enrolment Credential Keys: used to sign AT/EC requests;
- Authorization Ticket Keys: used to sign ITS messages.

Key destruction is implemented for all keys and related cryptographic material when are within the TOE boundary and are no longer needed. Keys inside the TOE are deallocated from TOE RAM after they have been used in the current operation by performing zeroization of the RAM area where the keys were temporarily stored. Exported keys are made permanently unavailable by destroying the master key.

1.3.3.2 Digital Signature Generation

The TOE generates digital signatures according to the ECDSA (Elliptic Curve Digital Signature Algorithm) scheme serving the VCS for data and entity authentication supporting ETSI standards **[TS 103 097]** and **[TS 102 941]**:

- Data integrity and origin authentication: an ITS message is signed by an AT private key to generate a proof of authenticity and integrity for the recipient
- Entity authentication: EC/AT requests are signed by Canonical/Enrolment Credential private key to authenticate the TOE to the Certification Entities (EA/AA).

1.3.3.3 User data ECIES encryption/decryption

When ITS message confidentiality is requested, the VCS generates a secret data encryption key, encrypts the message with the data encryption key and invokes ECIES encryption service from the V2X eHSM. The TOE receives as input parameter the recipient public key, key derivation and encoding parameters, a recipient information string, and the VCS data encryption key and uses ECIES (Elliptic Curve Integrated Encryption Scheme) for encryption of the data encryption key. The encrypted data encryption key, the authentication tag and the sender ephemeral public key are exported to the VCS

(see Figure 1). The corresponding decryption process is described in Figure 2. The algorithm, parameters and formats for ECIES are defined in **[IEEE1609.2]** and **[TS 103 097].**

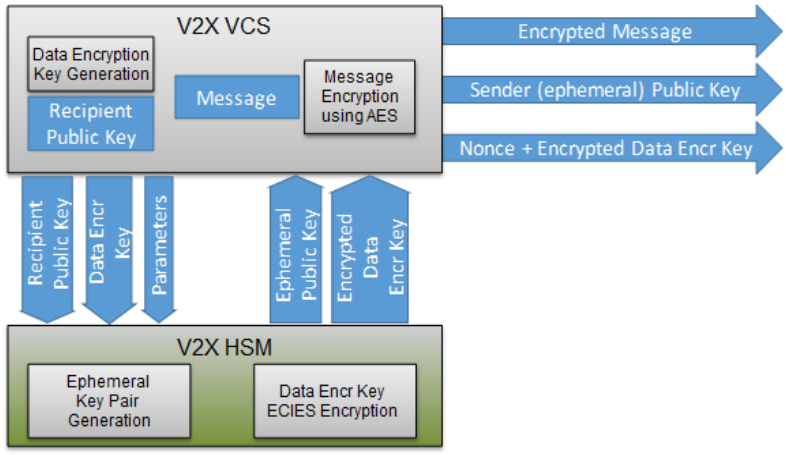


Figure 1. TOE input/output for ECIES message encryption

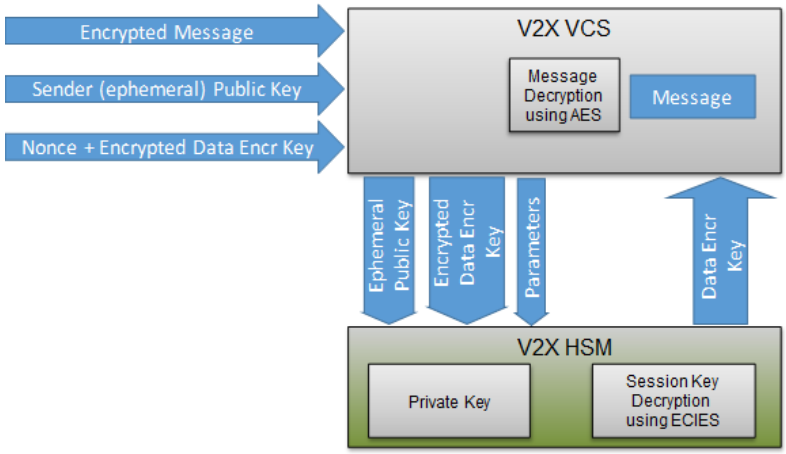


Figure 2. TOE input/output for ECIES message decryption

1.3.3.4 ECC key derivation

The TOE supports an ECC key derivation service required for the Butterfly Key Expansion mechanism as specified in **[IEEE 1609.2.1]** chapter "9.3 Butterfly key mechanism". Key derivation can be performed by using ECC private keys that were generated by the TOE or imported into the TOE.

The objective of the Butterfly Key expansion mechanisms is to generate an arbitrary number of Authorization Ticket private keys and to issue corresponding certificates. The derived private keys will be used for ECDSA signature generation and ECIES operations.

However, the TOE does not provide the whole mechanism. The mechanism implemented by the TOE are described below:

- Generate caterpillar private key
- Private key derivation with a caterpillar private key to produce a cocoon private key
- Private key derivation with a cocoon private key to produce a butterfly private key

The private key derivation performed by the TOE is limited to mathematic operations involving private key, that are modular additions and multiplications with arguments provided by VCS. The private key result of this operation is wrapped in a key blob that is exported to the VCS.

1.3.3.5 Random number generation

A random number generator is used by the TOE for internal usage of key generation and seeding its countermeasures and as an external service for the VCS.

1.3.3.6 Self-protection

The TOE provides a resistance to moderate attack potential based on hardware and firmware security measures allowing failure and physical attack resistance with preservation of a secure state.

1.3.4 TOE life-cycle

The TOE lifecycle is described in four phases: Development, manufacturing, platform integration, and operational usage. Because the TOE may support firmware update functionality, the TOE life-cycle defined in **[PP_V2XHSM]** distinguishes two cases:

- Case 1: Initial provisioning of the TOE hardware and firmware.
- Case 2: Firmware update of the TOE (only applicable if the functional package “Software Update Package”).

The main difference between both lifecycles is based on the existence of a firmware update feature. The TOE described in this ST is not able to perform firmware updates throughout its life cycle, so the description of TOE's life-cycle covered by this security target corresponds to **Case 1** of the two above.

The scope of this security target covers the phases 1 and 2 of the life cycle. These phases include all the necessary steps to move on the TOE in a final state which it can be integrated into the destination platform where it will carry out its normal operation.

The case 1 of the TOE life cycle can be summarized as follows:

- **TOE Development (Phase 1)**: this phase comprises the development of the TOE hardware design by ST Microelectronics and the TOE firmware design by Autotalks.
- **TOE Manufacturing and Delivery (Phase 2)**: this phase comprises the production of the integrated circuit, building the ROM of the TOE containing the TOE firmware, testing the wafer, delivery of the TOE (tested wafers) to the IC manufacturer.
- **Platform Integration (Phase 3):** during this phase, the TOE is packaged, integrated in the platform and delivered to the client of the platform integrator.
- **Operational Usage (Phase 4)**: in this phase, the TOE is prepared for operational usage and used in the environment of the end-user. The preparative procedures for operational usage include secure acceptance of the delivered TOE.

Application Note

The import or generation of the canonical key and other keys is performed in phase 3: Platform integration.

Authorization Ticket (Pseudonym Certificate) keys may be also imported or generated in phase 4: Operational Usage.

Finally, symmetric keys are generated and used by the TOE during platform integration and operational usage (phase 3 and 4) for protecting the private ECC keys.

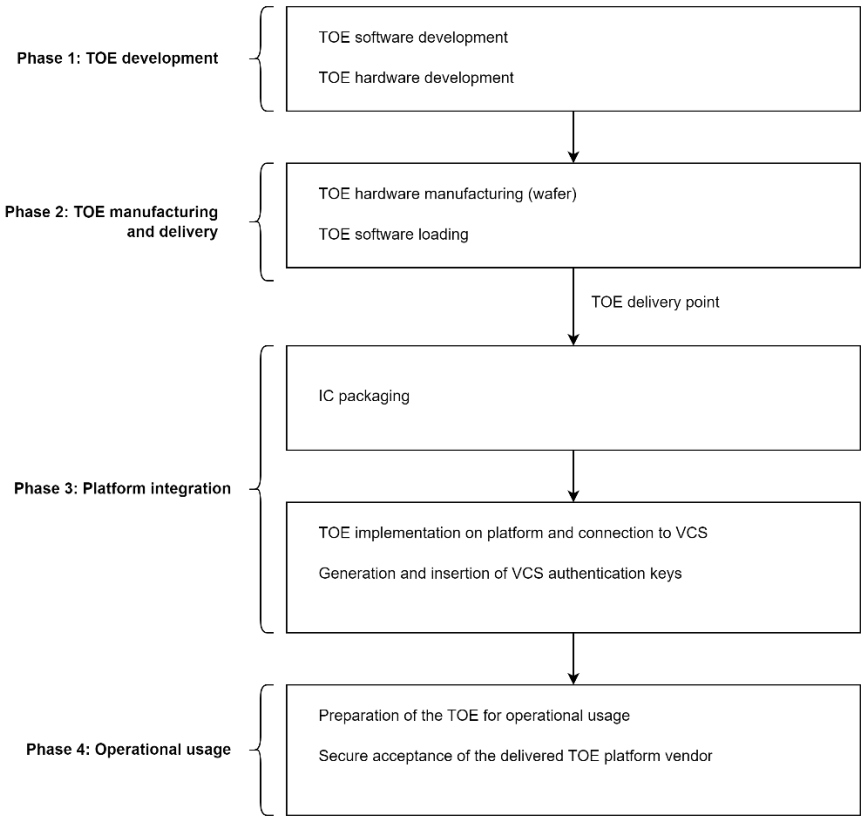


Figure 3. TOE life cycle for Case 1.

The current security target covers the following phases:

- Phase 1: development of the TOE
- Phase 2: TOE manufacturing and delivery. The scope covers the manufacturing of the TOE up to the delivery of the wafer to the IC manufacturer (cf. **[CC31R5P1]** paragraph 157).

The particular state of the TOE when delivered to the IC manufacturer as client of the TOE vendor depends on the vendor configuration options. Details on this configuration are provided for evaluation activities of families ALC_CMS and ALC_DEL.

The user guidance of the TOE vendor describes the requirements and general procedures to be followed by the supplier of the certified TOE to enable the end-user's acceptance of certified version and configuration of the delivered TOE (cf. element AGD_PRE.1.1C for details).

Application Note:

The phases 1 and 2 covered by this security target of the life-cycle of the TOE are carried out by the manufacturer (ST Microelectronics) together with Autotalks.

After the development of the mentioned parts, the TOE will be ready to be integrated in the VCS to perform the V2X communication.

1.3.5 Available Non-TOE Hardware/Software/Firmware

The TOE is an embedded hardware module with related firmware. The TOE is designed to be embedded in a SoC according to the architectures described in section 1.4.2. This SoC is intended to be part of larger systems like automotive control units.

The TOE's security services are very specific and well-defined; therefore, the TOE requires some operational environment elements to provide complimentary security services which contributes to mitigate the threats. These services are under control of the operational environment and are not enforced by the TOE.

The Non-TOE elements required by the TOE are:

- VCS's non-secure, non-volatile memory (NVM) where TSF data from the TOE is stored in an encrypted way.
- The SoC SECTON or CRATON2, and in particular, their elements in charge of enabling communication, control, power supply and self-protection for the TOE, in particular:
    - Cortex-M3 Processor.
    - Mailbox instance in Cortex-M3 Processor.
    - Cortex-A7 Processor.
    - Mailbox instance in Cortex-A7 Processor.
    - Reset line.
    - Main power line.
    - NIC400 interconnect bus.
    - Clock lines.
    - RAM shared memory.
    - Tamper detection line
    - Physical protection measures: metal layer, self-protection sensors (voltage, clock, temperature, power)

1.4 TOE Description

1.4.1 Introduction

This section describes the TOE architecture and boundary, TOE physical scope, TOE logical scope and TOE evaluated configuration.

1.4.2 TOE Architecture and Boundaries

The deployment option from those in **[PP_V2XHSM]** to which this Security Target is compliant is the option 2:

- Deployment option 2 (Figure 4) considers the V2X HSM placed in the same SoC. In this deployment the VCS – V2X HSM communication is secured by physical means.

according to the schematic view shown below:

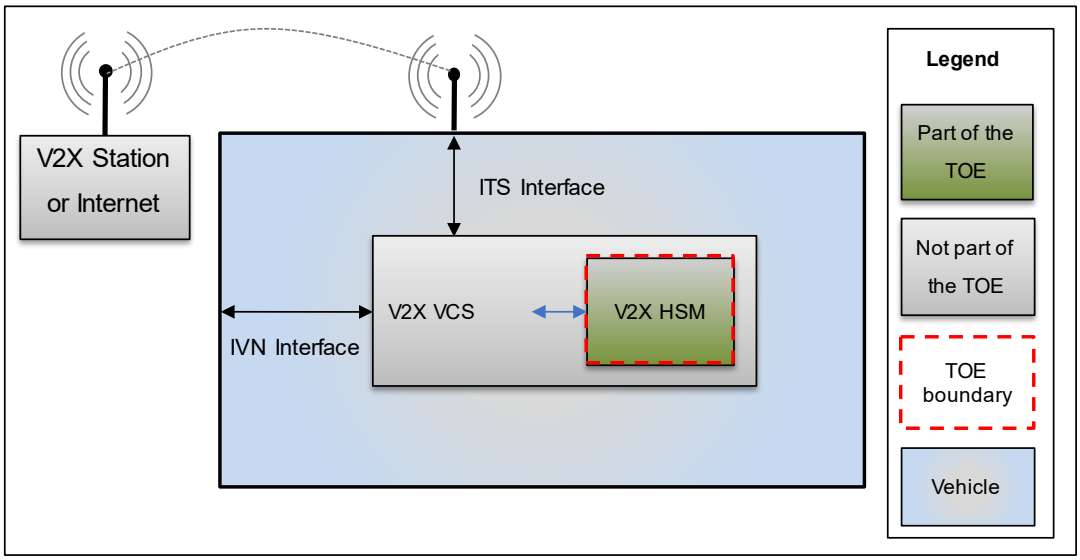


Figure 4. TOE system overview, integrated V2X eHSM

The TOE boundary is a tamper resistant hardware module including the firmware required for its functionality, which is embedded in the same silicon than the rest of the SoC.

The link (represented by the blue arrow) between the VCS (operational environment) and the TOE, as part of the same silicon, is protected by the physical security measures of the rest of the SoC (operational environment) which includes:

- Upper and lower silicon layers and metal layer protections
- Tamper detection sensors in the SoC including voltage, clock and temperature
- SoC encapsulation

The access to this link is therefore also protected by the operational environment with the aforementioned security measures.

The TOE is deployed in two different SoC versions:

CRATON2 architecture

In this architecture, depicted in Figure 5 below, the VCS is completely embedded in the SoC and runs on two CPUs inside the SaS domain, ARM Cortex-A7 and Cortex-M3. Each of the processors is physically connected to the TOE and can communicate with it directly.

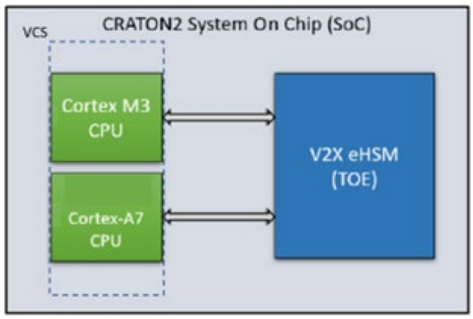


Figure 5 CRATON2 architecture

In this case, both communication links between the user (the VCS) and the TOE (the one link through M3 and the other through A7) are embedded as part of the SoC and are protected by the operational environment as described previously.

SECTON architecture

Depicted in Figure 6 below, is a reduced subset version of CRATON2 having only the Cortex-M3 CPU and is used for integration with an external host. In this architecture the VCS relies in a SoC-external component in the host side and in a SoC-internal component which is embedded in the SoC (the Cortex-M3).

Being that the channel between both VCS components do not belong to the same physical domain, both components are cryptographically linked.

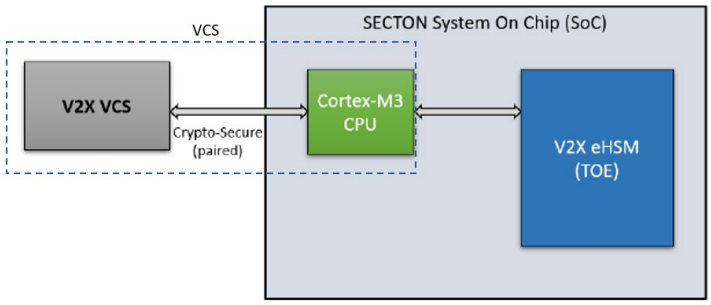


Figure 6 SECTON architecture

In this case, the only communication link between the user and the TOE is driven by Cortex-M3 (the external component of the VCS is not accessing this link), is completely included in the SoC and therefore is protected by the operational environment as described previously.

1.4.3 TOE Logical Scope

The TOE Logical scope comprises the following security functionality:

- V2X Key Management:
  - Generation
  - Secure import
  - Secure export
  - Destruction
- Cryptographic operations:
  - Digital signature generation
  - User data ECIES encryption/decryption
  - ECC key derivation
- Random number generation
- Self-protection

V2X Key management

The TOE handles key generation, secure import and export of ECC private keys and symmetric keys and their destruction.

- Key generation: The TOE supports the generation of the ECC key pairs used for ECDSA-based signature and ECIES encryption/decryption. The TOE also support the generation of AES keys (wrapping keys) for the secure export of data.
- Key derivation: The TOE implements a key derivation mechanism as specified in **[IEEE 1609.2.1]** chapter "9.3 Butterfly key mechanisms". The private key result of this operation is wrapped in a key blob which is exported to the VCS.
- Secure export: The ECC keys are wrapped inside an AES based cryptographically protected blob structure to be stored out of the TOE, in the VCS's non-volatile memory.
- Secure import: based in the Private Key Import (offline) package of **[PP_V2XHSM],** the ECC private keys protected in a blob structure can be securely imported into the TOE. When this key is imported, it is re-encrypted and authenticated as a regular blob and exported to the VCS and can be later on be used for ECDSA-based signature, ECIES decryption and key derivation.
- Key destruction: the ECC keys and wrapping keys under TOE control are destroyed when no longer needed or as a result of security violations.

Cryptographic operations

The TOE makes use of a suite of algorithms to perform the necessary cryptographic actions needed to provide its functionality. These cryptographic operations are described as follows:

- **ECDSA-based signature generation service**, as a service provided to the user by specific API, which supports NIST and Brainpool prime curves of different key sizes, according to **[FIPS 186-4]** standard.
- **ECIES encryption/decryption** is also provided by the TOE as a security service to the user by specific API, required for securing V2X communications.
- **AES encryption-decryption with authentication** as an internal service only, used for making possible the secure import/export of data.

Random number generation

The TOE includes a DRBG module seeded by a TRNG. This DRBG is used internally for generation of ECC private keys, symmetric keys, ECC key derivation as well as for seeding countermeasures mechanisms in the cryptographic engines. The DRBG can be also accessed from the outside for requesting random numbers through a specific API.

Self-protection

The TOE includes self-protection mechanisms which monitors the TSF and enable different actions depending on its condition:

a) The TOE enters panic mode if a fault is detected during operation due to unexpected hardware behaviour (e.g., triggered by an unexpected error generated by the crypto engines), or due to self-test failure. In panic mode, the TOE zeroizes the RAM, halts the Cortex-M0 processor and signals panic mode to the SaS through the hardware mailbox.

b) The TOE has a communication interface intended to receive tamper signals from the SaS. When the TOE receives a tamper signal, it starts a destruction sequence that leaves the TOE in an unusable status. The destruction sequence consists of:
   - Performing the panic mode sequence: RAM zeroization, CPU halting and panic mode signalling to the SaS.
   - Destruction of the master key (CSK) making the decryption of keys impossible.

c) The TOE runs a set of self-tests during boot, entering a secure failure state if failure occurs. Self-tests are also run periodically during execution, on-demand by invoking specific APIs, and upon specific conditions.

d) The TOE includes internal countermeasures for protecting against physical attacks: redundancy bits, random delays, low voltage detection, dummy operations and masking of critical security parameters.

e) The TOE hardware design also contributes to enforce self-protection of the TSF. As part of the SoC physical domain, the high integration of the TOE and the use of the same power domain than the rest of the modules in the SoC, requires the attacker to use of very high-resolution detection equipment to distinguish the power used by the TSF cryptographic operations from the whole power consumed by the rest of the SoC.

Any other secure service accessible by the user different than the ones specified in this section shall be considered as Non-TOE.

1.4.3.1 Non-TOE Security Features

TSF functionality can be accessed from the TSFIs specified in [FSP]. There is no access to the TOE that can be considered as Non-TOE.

1.4.4 TOE Physical Scope

The physical scope of the TOE is comprised of:

- The TOE Hardware
- The TOE Firmware
- The TOE guides and documentation

The TOE is a hardware module integrated in a SoC of one of the architectures described in section 1.4.2.

At hardware level, the TOE is composed of a series of parts that make it possible to provide security functionalities in an efficient manner and are listed below:

- **CPU (Cortex-M0)**. A dedicated core reserved for the execution of security-related routines. This CPU allows limited communication through AHB Lite bus and the mailbox. In addition, the CPU manages the TOE to parse the clients' requests, dispatch the request to the relevant component of the TOE.
- **Mailbox:** hardware mailbox that allows SaS to send interrupts to eHSM and the eHSM to send interrupts to SaS. This component is used passing signals between the eHSM and the Cortex-M3 and Cortex-A7.
- **RAM memory**. The RAM memory is used for storing request parameters from the TOE user, intermediate computation results and the stack of the TOE program.
- **ROM memory**. The purpose of this memory is to store the TOE's firmware and constant data.
- **OTP memory**. The TOE includes a One Time Programmable (OTP) memory where the permanent data is stored. This memory includes protection measures due to the relevance of the information stored. This protection consists of a specific OTP's firewall, a single error correction and redundancy for integrity protection of the data. Multiple configuration and life-cycle flags are stored in this memory.
- **C3 Crypto processor**. This subsystem is a programmable HW crypto accelerator and whose purpose is to manage all those cryptographic operations of the TOE, with the aim of providing the appropriate parameters and conditions to carry out all operations efficiently.
- **AHBLite bus.** This bus is used internally in the TOE to interconnect the TOE internal components and to connect the TOE with the external access to the SoC NIC400 Interconnect bus.

The TOE also includes external physical interfaces to connect the TOE to external components:

- Interface to SoC NIC400 Interconnect bus to allow access to SoC shared memory for V2X eHSM communications.
- Clock, power, reset lines
- An anti-tamper signal line.
- Mailbox

The TOE firmware is the part that implements the logical TSF. The firmware is programmed in the ROM memory inside the eHSM.

The TOE guides are presented in the table below:

Document	Version	Format	Delivery Method
SECTON/CRATON2 Embedded V2X HSM CUT3.1 (TOE)	CUT 3.1: - Firmware version: 0xC1010400 - Hardware version: Wafer VA83D	Embedded in the SoC delivered in wafer form	Trusted courier
SECTON-CRATON2 Embedded V2X HSM CUT3.1 Functional Specification-FSP	1.4	PDF Document	Encrypted file
SECTON-CRATON2 Embedded V2X HSM CUT3.1 Operational User Guidance-OPE	0.9	PDF Document	Encrypted file
SECTON-CRATON2 Embedded V2X HSM CUT3.1 Preparational Guidance-PRE	1.3	PDF Document	Encrypted file

Table 1. TOE Guidance documents

1.4.5 TOE Evaluated Configuration

Logical configuration

The eHSM supports multiple logical configurations defined by a wide set of parameters (flags) that can modify the behaviour of the security services of the TOE.

The evaluated configuration of the TOE in this Security Target consists of a fixed-value configuration of these flags to enable the TOE with the security services described along this Security Target.

These flags are configured by the firmware deployed by Autotalks and no user can change them during operational mode.

Physical configuration

Regarding the physical configuration the TOE is delivered inside the SoC, which is provided in wafer form to the TOE integrator. The SoC containing the TOE can be provided in the versions SECTON or CRATON2 as described in section 1.4.2, having the same TOE internal architecture and configuration.

All the security mechanisms provided by the TOE and described in this Security Target are always enforced in both configurations and they do not require any specific action by the user.

2 Conformance Claims

This Security Target and the TOE described are in accordance with the requirements of Common Criteria 3.1R5.

This Security Target claims conformance with the following parts of Common Criteria:

- Conformance with **[CC31R5P1].**
- Conformance with **[CC31R5P2]** extended.
- Conformance with **[CC31R5P3].**

The methodology to be used for the evaluation is described in the "Common Evaluation Methodology" of the Common Criteria standard of April 2017, version 3.1 revision 5 with an evaluation assurance level of EAL4 + AVA_VAN.4, ALC_FLR.1.

2.1 PP Conformance Claims

This Security Target claims strict conformance to Protection Profile Car2Car Communication Consortium HSM (**[PP_V2XHSM]).**

2.2 Package Conformance Claims

This Security Target claims conformance with the following packages of **[PP_V2XHSM]:**

- Private Key Import (offline) Package
- Key Derivation Package

This Security Target claims conformance to the following assurance package of **[CC31R5P3]:**

- EAL4 + AVA_VAN.4, ALC_FLR.1

3 Security Problem Definition

3.1 Introduction

The security problem definition described below includes threats, organisational security policies and security usage assumptions.

3.2 Assets

Asset	Description
ECC private keys[1]	Cryptographic keys used exclusively by the TOE. In eHSM context, several types of ECC keys are handled: • ECC private keys used to perform digital signature operations; • ECC private keys used in ECIES; In V2X VCS context, ECDSA private keys are: • Canonical Key: used to sign EC requests; • Enrolment Credential Keys: used to sign AT requests; • Authorization Ticket Keys: used to sign ITS messages. The Protection Profile **[PP_V2XHSM]** does not enforce handling of key types by TOE. These assets must be protected in confidentiality and integrity for private ECC.
VCS data	User data exchanged between TOE and the VCS. In V2X VCS context, VCS data can be: • Representation of parts of EC/AT requests or ITS information provided to the V2X eHSM to be signed; • Data encryption key (symmetric) provided to the V2X eHSM to be encrypted/decrypted (ECIES); • Recipient public key and parameters provided to the V2X eHSM for ECIES encryption; • Sender (ephemeral) public key and parameters provided to the V2X eHSM for ECIES decryption; • Public keys returned by TOE corresponding to ECC private keys generated by the TOE; • Random number generated by the TOE provided to VCS. User data must be protected at minimum in integrity. Furthermore, confidentiality protection is required for data to be ECIES encrypted/decrypted and for random numbers. The protections are needed during communication and while in operation by the TOE.
Secure Services	The secure services provided by the TOE are listed below: • Random Number Generation

	• Key generation • Key destruction • Key import • Key export • ECDSA Digital signature generation • ECIES encryption • ECIES decryption • Key derivation • Self-protection Secure services must be protected in runtime integrity.
eHSM firmware	Encoded instructions that regulate the behavior of the TOE. eHSM firmware must be protected in integrity.
Wrapping keys[2]	Cryptographic keys[3] used to wrap generated and imported keys to ensure their confidentiality, authenticity, and integrity. This asset is protected in confidentiality, authenticity and integrity.

Table 2. Assets to be protected by the TOE

Application notes

[1] For the cryptographic keys the integrity only covers changes controlled by an attacker leading to knowledge of private keys, or modification of public key to value chosen by the attacker.

[2] Same threats as for ECC private keys apply to Wrapping keys.

[3] According to **[PP_V2XHSM]**, wrapping keys are only used to protect imported packages to the TOE. However, for this evaluation the term 'wrapping key' refers also to the keys involved in the storage of the blobs in the external memory.

3.3 Users

The eHSM only considers one user according to the *Table 3. Users' description*. This user is the only one that can invoke the security services from the TOE. Therefore, these services cannot be activated by anyone else.

The TOE is part of a more complex system in which it provides its security services to enable the VCS to stablish secure communication according with the V2X protocol. In this scenario, the TOE user is the VCS through Cortex-M3 and Cortex-A7 processors, which are responsible to perform the V2X package transmission. It is shown in Figure 5 and Figure 6.

User	Description
VCS through Cortex-M3 or Cortex-A7.	User invoking Secure Services of the TOE.

Table 3. Users' description

3.4 Threat Agents

Two main types of attackers have been identified, both attacker types have moderate attack potential.

Threat Agent	Description
Local attacker	Attacker with physical access to the TOE, such attacker does not have an authorized access to the TOE services (other than through the VCS during operation of the vehicle). Local attacker can run hardware or firmware attacks through physical or logical TOE interfaces.
Remote attacker	Attacker with access (authorized or not) through the VCS; such attacker has an authorized access to the TOE services by means of VCS. Remote attacker can run hardware or firmware attacks through logical TOE interfaces only.

Table 4. Threat agents

3.5 Threats

Threats are described by an adverse action performed by defined threat agents on the assets that the TOE has to protect.

Attackers in V2X networks will have two objectives in the final V2X context:

- Be able to track a vehicle.
- Cause safety hazardous situation.

The V2X eHSM provides supporting functionalities to prevent such risks.

The threats against the TOE according to Table 5 are identified. In this table, the generic term “attacker” is used to cover both local and remote type of attacker (see previous section). Attacks on data can be “direct” or using existing services.

Threat	Description	Asset/protection
T.KEY_REPLACE	An attacker is able to replace ECC private key (stored inside the TOE or exported outside the TOE within an encrypted and authenticated blob structure that can be decrypted and authenticated by the TOE only.) by one he knows (e.g., generated by him, or taking a weak value) without being detected by the TOE. In V2X context, the attacker will be able to: - track the victim vehicle (key known); - request a certificate for the public key and then sign himself (out of TOE) wrong information (on behalf of the victim or of himself). Note: The integrity only covers changes controlled by an attacker leading to knowledge of private keys, or modification of public key to value chosen by the attacker.	ECC private keys / Wrapping keys / Integrity

	Compromise of the integrity of keys leading to unavailability of the device is not in the scope of the Protection Profile to which this ST claims conformance.	
T.KEY_DISCLOSE	An attacker is able to disclose ECC private key (stored inside the TOE or exported outside the TOE within an encrypted and authenticated blob structure). In V2X context, the attacker will be able to: -track the victim vehicle (key known); -sign himself (out of TOE) wrong information (on behalf of the victim or himself).	ECC private keys / Wrapping keys / Confidentiality
T.SW_TAMPER	An attacker is able to modify the eHSM firmware (implemented in ROM); he then has a partial control of the TOE behaviour and potentially on assets. In V2X context, various exploitations will be possible depending on the modifications (see impacts in other threats as examples).	eHSM firmware/ Integrity
T.SRV_MALFUNCTION	An attacker may take advantage of a malfunction of the Secure Services. This may affect any asset and could result in any of the other threats.	Secure Services / Integrity
T.SW_REPLACE	An attacker is able to directly replace the eHSM firmware (implemented in ROM); he then has the full control on TOE behaviour and then on assets. In V2X context, all exploitation will be possible (see impacts in other threats as examples).	eHSM firmware / Integrity
T.VCS_DATA_MODIF	An attacker is able to modify VCS data during communication and when processed by the TOE. In V2X context, the attacker will then be able to make sign wrong information; if modifications are controlled so the message can be interpreted by receivers, it can provoke an undesired reaction of the vehicle; if modifications are not controlled and cannot be interpreted, this could at least make receivers consume resources unduly or provoke unexpected reactions of receiver devices (e.g., crash).	VCS data / Integrity
T.VCS_DATA_DISCLOSE	An attacker is able to disclose VCS data during communication and when processed by the TOE when confidentiality has been requested by User. In V2X context, when data is the data encryption key the attacker will then be able to decrypt data exchanged between VCS and PKI. The exchanged data comprises certificate signing requests, including long term identity of the vehicle, as well as authorization tickets. If this information is disclosed the privacy of the vehicle it compromised. When data is random number used for key generation by the VCS, the attacker will then be able to disclose the data encryption key.	VCS data / Confidentiality

Table 5. Threats against the TOE

3.6 Organizational Security Policies

Organisational Security Policies, OSPs, are defined according to Table 6

Organizational Security Policy	Description
PSIGNATURE_GENERATION	The TOE shall be able to generate ECDSA digital signatures.
P.KEY_GENERATION	The TOE shall be able to generate ECC asymmetric key pairs for ECDSA and ECIES operations.
P.ECIES	The TOE shall be able to encrypt and decrypt VCS data according to ECIES.
P.RNG	The TOE is required to generate random numbers that meet specified quality metric, for use by the TOE itself and the VCS. These random numbers shall be suitable for use as keys, authentication/authorization data or seed data for another random number generator.
PSECURE_COMMUNICATION	The TOE environment must implement protection for integrity and confidentiality if required, of VCS data when exchanged between the TOE and the VCS.
P.SRV_ACCESS	Access to the V2X eHSM services shall be restricted to the VCS only.
P.PRIVKEY_IMPORT_AE	The TOE shall be able to import authenticity, integrity and confidentiality protected ECC private keys generated externally.
P.KEY_DERIVE	The TOE and the operational environment together shall implement or support key derivation following **[IEEE 1609.2.1]** chapter "9.3 Butterfly key mechanisms".

Table 6. Organisation Security Policies

3.7 Assumptions

Assumptions on the TOE operational environment are made according to Table 7.

Assumption	Description
A.INTEGRATION	It is assumed that appropriate technical and/or organizational security measures in the Platform Integration (Phase 3) of the initial provisioning of the TOE hardware and firmware (Case 1) in order to guarantee for the confidentiality, integrity and authenticity of the assets of the TOE.

Table 7. Assumptions on the TOE environment

4 Security Objectives

4.1 Introduction

The statement of security objectives defines the security objectives for the TOE and its environment. The security objectives intend to address all security environment aspects identified. The security objectives reflect the stated intent and are suitable to counter all identified threats and cover all identified organisational security policies and assumptions. The following categories of objectives are identified:

- The security objectives for the TOE shall be clearly stated and traced back to aspects of identified threats to be countered by the TOE and/or organisational security policies to be met by the TOE.
- The security objectives for the environment shall be clearly stated and traced back to aspects of identified threats countered by the TOE environment, organisational security policies or assumptions.

4.2 Security objectives for the TOE

The following security objectives for the TOE (OT) are defined.

Security Objective	Description
OT.PRIVKEY_ACCESS	The TOE shall ensure that private keys can only be used through V2X services to which access is restricted to User and cannot be retrieved out of the TOE in a form allowing usage outside of the TOE.
OT.SIGNATURE_GENERATION	The TOE shall be able to generate ECDSA digital signatures on VCS data.
OT.KEY_MANAGEMENT	The TOE shall be able to generate, store, and protect ECC asymmetric keys for ECDSA and ECIES operations and symmetric keys.
OT.ECIES	The TOE shall be able to encrypt and decrypt VCS data according to ECIES (as described in section 1.3.3.3).
OT.TOE_SELF-PROTECTION	The TOE shall be able to protect itself and its assets from manipulation including physical and firmware tampering.
OT.RNG	Random numbers generated shall meet a defined quality metric in order to ensure that random numbers are not predictable and have sufficient entropy. For security operations, e.g., key generation, high quality random numbers are required.
OT.VCS_DATA	The TOE shall implement security measures to prevent alterations, and disclosure when confidentiality is requested, of received user data stored and processed in the TOE.
OT.PRIVKEY_IMPORT_AE	The TOE shall be able to import authenticity, integrity and confidentiality protected ECC private keys generated externally.
OT.KEY_DERIVE	The TOE shall support private key derivation following **[IEEE 1609.2.1]** chapter “9.3 Butterfly key mechanism”. The TOE shall provide inputs for key

	derivation. These inputs shall be protected in authenticity, integrity and confidentiality by the TOE.

Table 8. Security objectives for the TOE

4.3 Security Objectives for Operational Environment

The following security objectives for the Environment (OE) are defined.

Security Objective	Description
OE.SECURE_COMMUNICATION	The TOE operational environment must implement protections for integrity and confidentiality of VCS data when exchanged between the TOE and the VCS in accordance with protections specified in chapter 3.2 (asset definition). This protection can be limited to physical protection.
OE.SRV_ACCESS	The TOE environment must implement security measures to restrict V2X eHSM services access to the VCS only. This protection can be limited to physical protection.
OE.INTEGRATION	Appropriate technical and/or organizational security measures shall be in place in the Platform Integration (Phase 3) in order to guarantee the confidentiality, integrity and authenticity of the assets of the TOE in accordance with protections specified in chapter 3.2 (asset definition).
OE.KEY_MANAGEMENT	In case a key pair is generated outside the TOE to be then imported, the environment shall ensure that key pair are securely managed: - Key generation service shall be provided to authorized users only; - Key generation shall be performed in accordance with **[FIPS 186-4], [RFC 5639]**; Confidentiality of private key shall be ensured while outside the TOE.
OE.KEY_DERIVE	The operational environment shall provide inputs for key derivation following **[IEEE 1609.2.1]** chapter “9.3 Butterfly key mechanism”.

Table 9. Security objectives for the TOE operational environment

4.4 Security Objectives Rationale

4.4.1 Security Objectives Coverage

This section provides tracings of the security objectives for the TOE to threats, OSPs, and assumptions.

	OT.PRIVKEY_ACCESS	OT.SIGNATURE_GENERATION	OT.KEY_MANAGEMENT	OT.ECIES	OT.TOE_SELF-PROTECTION	OT.RNG	OT.VCS_DATA	OT.PRIVKEY_IMPORT_AE	OT.KEY_DERIVE	OE.SECURE_COMMUNICATIO	OE.SRV_ACCESS	OE.INTEGRATION	OE.KEY_MANAGEMENT	OE.KEY_DERIVE
T.KEY_REPLACE	X		X		X									
T.KEY_DISCLOSE	X		X		X									
T.SW_TAMPER					X									
T.SRV_MALFUNCTION					X									
T.SW_REPLACE					X									
T.VCS_DATA_MODIF					X		X			X				
T.VCS_DATA_DISCLOSE					X		X			X				
P.SIGNATURE_GENERATION		X				X								
P.KEY_GENERATION			X			X								
P.ECIES				X		X								
P.RNG						X								
P.SECURE_COMMUNICATION										X				
P.SRV_ACCESS											X			
P.PRIVKEY_IMPORT_AE								X					X	
P.KEY_DERIVE									X					X
A.INTEGRATION												X		

Table 10. Security Objectives vs Security Problem Definition

4.4.2 Security Objectives Sufficiency

The following rationale provides justification that:

- the security objectives for the environment are suitable to cover each individual assumption or threat to the environment;

- each security objective for the environment that traces back to a threat or an assumption about the environment of use.

Threats/OSP/Assumption	Objective	Rationale
T.KEY_REPLACE	OT.KEY_MANAGEMENT OT.PRIVKEY_ACCESS OT.TOE_SELF-PROTECTION	Once generated, private keys are securely exported outside the TOE in an encrypted and authenticated blob structure. Modification of this blob will result an authentication error during their decryption and authentication when imported into the TOE. Logical write access to private keys stored in the TOE is only possible through Secure Services to which access is restricted to User. The TOE is protected from physical and firmware tampering –among others– replacement of keys by circumventing access control.
T.KEY_DISCLOSE	OT.KEY_MANAGEMENT OT.PRIVKEY_ACCESS OT.TOE_SELF-PROTECTION	Once generated, private keys are securely exported outside the TOE in an encrypted and authenticated blob structure that can be accessed only by the same TOE. The TOE is protected from physical and software tampering to prevent – among others – disclosure of keys by circumventing access control.
T.SW_TAMPER	OT.TOE_SELF-PROTECTION	The TOE is protected from physical and firmware tampering; thus, parts of the TOE cannot be modified.
T.SRV_MALFUNCTION	OT.TOE_SELF-PROTECTION	The TOE is protected from physical and firmware tampering protecting against any malfunction.
T.SW_REPLACE	OT.TOE_SELF-PROTECTION	The TOE is protected from physical and firmware tampering, thus the eHSM firmware cannot be illegally replaced.
T.VCS_DATA_MODIF	OT.VCS_DATA	The TOE is protected from physical and firmware tampering protecting

Threats/OSP/Assumption	Objective	Rationale
	OT.TOE_SELF-PROTECTION OE.SECURE_COMMUNICATION	against data illegal modification of – among others– VCS data processed inside the TOE. The integrity of VCS data during communication is protected by the Operational Environment. The VCS data have integrity protections when stored or processed by the TOE.
T.VCS_DATA_DISCLOSE	OT.VCS_DATA OT.TOE_SELF-PROTECTION OE.SECURE_COMMUNICATION	The VCS data have confidentiality protections when stored or processed by the TOE. The TOE is protected from physical and firmware tampering protecting against any data illegal disclosure of – among others – VCS Data processed inside the TOE. The confidentiality of VCS data during communication is protected by the Operational Environment.
P.SIGNATURE_GENERATION	OT.SIGNATURE_GENERATION OT.RNG	OT.SIGNATURE_GENERATION is rephrasing the OSP. The quality of the random numbers required for signatures is ensured by the TOE.
P.KEY_GENERATION	OT.KEY_MANAGEMENT OT.RNG	OT.KEY_MANAGEMENT is rephrasing the OSP. Key generation inside the TOE is based on a random number generation ensuring randomness quality.
P.ECIES	OT.ECIES OT.RNG	OT.ECIES is rephrasing the OSP. The quality of the random numbers required for encryption based on ECIES is ensured by the TOE.
P.RNG	OT.RNG	OT.RNG is rephrasing the OSP.

Threats/OSP/Assumption	Objective	Rationale
P.SECURE_COMMUNICATION	OE.SECURE_COMMUNICATION	OE.SECURE_COMMUNICATION is rephrasing the OSP.
P.SRV_ACCESS	OE.SRV_ACCESS	OE.SRV_ACCESS is rephrasing the OSP.
P.PRIVKEY_IMPORT_AE	OT.PRIVKEY_IMPORT_AE OE.KEY_MANAGEMENT	The private key import feature is addressed by the TOE through the OT.PRIVKEY_IMPORT_AE. Moreover, to maintain the security of the Secure Services, the external key generation must also securely handle the key generation and handling while outside of the TOE through a blob structure. In case a key pair is generated outside the TOE to be then imported in a blob structure, the environment shall ensure that key pair are securely managed. Confidentiality of private key shall be ensured while outside the TOE (through a blob structure). OE.KEY_MANAGEMENT.
P.KEY_DERIVE	OT.KEY_DERIVE OE.KEY_DERIVE	The P.KEY_DERIVE policy is directly covered by OT.KEY_DERIVE. The generation of the private key used as an input for the TOE must be securely handled by the environment which is covered by OE.KEY_DERIVE. The private key derived by this service is encrypted and authenticated inside a blob structure to be exported outside the TOE.
A.INTEGRATION	OE.INTEGRATION	OE.INTEGRATION is directly covering the assumption.

Table 11. Security objectives sufficiency

5 Extended Components Definition

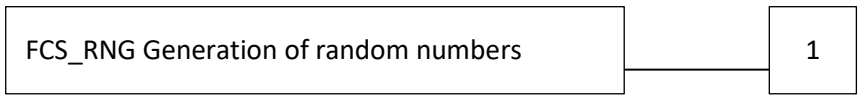
5.1 Definition of the Family FCS_RNG

To define the IT security functional requirements of the TOE an additional family (FCS_RNG) of the Class FCS (Cryptographic Support) is defined here. This extended family FCS_RNG describes an SFR for random number generation used for cryptographic purposes.

Family Behaviour

This family defines quality requirements for the generation of random numbers, which are intended to be used for cryptographic purposes.

Component Levelling



FCS_RNG.1 Generation of random numbers requires that the random number generator implements defined security capabilities and the random numbers meet a defined quality metric.

Management

FCS_RNG.1 There are no management activities foreseen.

Audit

FCS_RNG.1 There are no actions defined to be auditable.

FCS_RNG.1 Random number generation

FCS_RNG.1	Random number generation
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FCS_RNG.1.1	The TSF shall provide a [selection: *physical, non-physical true, deterministic, hybrid physical, hybrid deterministic*] random number generator that implements: [assignment: *list of security capabilities*].
FCS_RNG.1.2	The TSF shall provide random numbers that meet [assignment: *a defined quality metric*].

5.2 FCS_CKM.5 (Cryptographic key derivation)

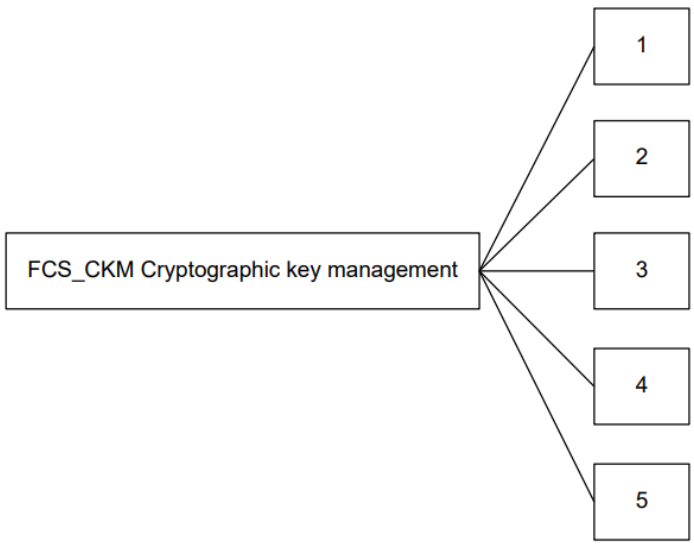
This extended component is based on the definition from **[PP_V2XHSM]**.

Family Behaviour

Cryptographic keys must be managed throughout their life cycle. This family is intended to support that lifecycle and consequently defines requirements for the following activities: cryptographic key generation, cryptographic key distribution, cryptographic key access and cryptographic key destruction. This family should be included whenever there are functional requirements for the management of cryptographic keys.

The extended component FCS_CKM.5 defines key derivation as process by which one or more keys are calculated from either a pre-shared key or a shared secret and other information. Key derivation is the deterministic repeatable process by which one or more keys are calculated from both a pre-shared key or shared secret, and other information, while key generation required by FCS_CKM.1 uses internal random numbers.

Component Levelling



FCS_CKM.5 Cryptographic key derivation requires the TOE to provide key derivation which can be based on an assigned standard.

Management

FCS_CKM.5 There are no management activities foreseen

Audit

FCS_CKM.5 The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:

a) Minimal: Success and failure of the activity.
b) Basic: The object attribute(s), and object value(s) excluding any sensitive information (e.g. secret or private keys).

FCS_CKM.5 Cryptographic key derivation

Hierarchical to: No other components.

Dependencies: [FCS_CKM.2 Cryptographic key distribution,

or FCS_COP.1 Cryptographic operation]

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.5.1 The TSF shall support derivation of cryptographic keys [assignment: key type] from [assignment: input parameters] in accordance with a specified cryptographic key derivation algorithm [assignment: cryptographic key derivation algorithm] and specified cryptographic key sizes [assignment: cryptographic key sizes] that meet the following: [assignment: list of standards].

6 Security Requirements

6.1 Definitions

6.1.1 Formatting Conventions

This section defines the Security functional requirements (SFRs) and the Security assurance requirements (SARs) that fulfil the TOE.

To be consistent with **[PP_V2XHSM]**, the conventions specified in its section 7.1.1 have been followed for all SFRs directly taken from the protection profile.

For those SFRs that have been modified from those in the PP, iterated, added or refined, the conventions below have been followed:

- Assignments. They appear between square brackets. The word “assignment” is maintained and the resolution is presented in ***boldface, italic and blue color.***
- Selections. They appear between square brackets. The word “selection” is maintained and the resolution is presented in ***boldface, italic and blue color.***
- Iterations. It includes a “iteration counter” in square brackets “()” in the title. A generic description of the requirement and a table with the information of each iteration is provided. Example: FCS_COP.1 (X iterations).
- Refinements: the text where the refinement has been done is shown ***bold, italic, and light red color.*** Where part of the content of a SFR component has been removed, the removed text is shown in ~~***bold, italic, light red color and crossed out.***~~
- The components have been copied directly from the Protection Profile. The operations made in the Protection Profile with respect CC part 2 have been highlighted **in bold** as is done in the Protection Profile.

6.1.2 Subjects, objects and security attributes

The following table defines subjects, objects and information which will be used in security functional requirements.

Subject/Object	Comments
S.User	Subject acting on behalf of the VCS through Cortex-M3 or Cortex-A7.
O.PrivateKey	ECC private keys[3]
SA.BlobType	Blob type related to the blob[4]

Table 12. Subject/Objects and security attributes

Application notes

[3] The keys are not stored within the TOE. Instead, they are imported into the eHSM in a blob structure whenever the keys are required for TOE operation.

[4] The values of the security attributes for the objects are set by the user for key generation and identified for key import. They are kept by the TSF during the requested operation and there is no external interface for the user to change, query, modify or delete.

6.1.3 Operations

The following table defines operations which will be used in security functional requirements.

Operations	Description
OP.KeyPair_Gen	ECC key pair creation
OP.RNG	Random number generation
OP.Signature	ECDSA signature generation
OP. EncDec	ECIES encryption/decryption
OP.Import	ECC private key import
OP.Key_derive	Key Derivation

Table 13. Definition of operations

6.1.4 Security Functional Policies

The following sections defines security functional policies which will be used in security functional requirements.

6.1.4.1 Private Key Access Control SFP

The TOE shall enforce this SFP to forbid the direct access to ECC private keys. The access to ECC private keys is allowed only via Secure Services. No user authentication, nor role management is required to be performed by the TOE, as this is handled by operational environment, see OE.SRV_ACCESS.

6.1.4.2 Private Key Import PCK SFP

The TOE enforces this SFP to securely manage O.PrivateKey object during OP.Import operation.

6.2 Common Generic Security Functional Requirements

The SFRs stated in this section are met by the TOE.

6.2.1 FCS: Cryptographic support

6.2.1.1 FCS_CKM.1 (two iterations) Cryptographic key generation

FCS_CKM.1.1/(id) The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm ***according to Table 14*** and specified cryptographic key sizes ***according to Table 14*** that meet the following: ***according to Table 14***.

id	Key generation algorithm	Key size	Standard
ECC	**ECC Key Pair Generation**	**256 bits [assignment: 384 bits]**	**[FIPS 186-4 Appendix B]**
AES [5]	***Symmetric key generation***	***256 bits***	***[SP800-133]***

Table 14 FCS_CKM.1 algorithm, key size and standards.

Application note

[5] AES key is used for encapsulating internally generated blobs used for exporting keys.

6.2.1.2 FCS_CKM.4: Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **[assignment: zeroization]** that meets the following: **[assignment: FIPS 140-2 Level 3]**.

Application Note

The key destruction covers the destruction of the following keys described in FCS_CKM.1 when are within the TOE boundaries:

- ECC keys
- Wrapping keys

6.2.1.3 FCS_CKM.5: Cryptographic key derivation

FCS_CKM.5.1 The TSF shall support derivation of cryptographic keys **ECC private key** from **an initial ECC private key** in accordance with a specified cryptographic key derivation algorithm **private key derivation mechanisms** and specified cryptographic key sizes **size of the initial ECC private key** that meet the following: **[IEEE 1609.2.1] chapter "9.3 Butterfly key mechanisms"**

Application Note

Support means mathematic operations involving private key, that are modular additions and multiplications with arguments provided by VCS, as described in 1.3.3.4.

6.2.1.4 FCS_RNG.1: Random number generation

FCS_RNG.1.1 The TSF shall provide a **[selection: deterministic]** random number generator that implements: **[assignment: a Deterministic Random Bit Generator (HMAC_DRBG) conformant to NIST [SP 800-90A], seeded by an internal TRNG)**

FCS_RNG.1.2 The TSF shall provide random numbers that meet **[assignment: independent bits with Shannon entropy of greater than 7.9999 bits per octet on a 2^{24} bit data-set]**.

6.2.1.5 FCS_COP.1 (six iterations): Cryptographic operations

FCS_COP.1.1/(Id) The TSF shall perform **the operations according to Table 15** in accordance with a specified cryptographic algorithm **according to Table 15** and cryptographic key sizes **according to Table 15** that meet the following: **according to Table 15**.

Id	Operation	Algorithm	Key length	Standard
ECDSA[6]	**Digital signature generation**	**ECDSA with NIST and Brainpool prime curves**	**256 bits, [assignment: 384 bits]**	**For algorithm: [FIPS 186-4] For curves: [FIPS-186-4] [RFC 5639]**
ECIES_ENC	**ECIES Encryption**	**ECIES with NIST and Brainpool prime curves**	**256 bits, [assignment: 384 bits]**	**For algorithm: [IEEE 1363a] For curves: [FIPS 186-4] [RFC 5639]**
ECIES_DEC	**ECIES Decryption**	**ECIES with NIST and Brainpool prime curves**	**256 bits, [assignment: 384 bits]**	**For algorithm: [IEEE 1363a] For curves: [FIPS 186-4] [RFC 5639]**

Import_Ver	**Verification of authenticity and integrity**	***AES-CCM mode***	***256 bits***	***[SP800-38C]***
Import_Dec	**Decryption**	***AES-CCM mode*** ***AES-CTR mode***	***256 bits***	***[SP800-38C]*** ***[SP800-38A]***
AES	***Authentication*** ***Encryption***	***AES-CCM mode with*** ***AES-CTR mode***	***256 bits***	***[SP800-38C]*** ***[SP800-38A]***

Table 15. FCS_COP.1 operations and key sizes

Application note

[6] Hash digest can be computed externally or internally. This operation is included in the defined ECDSA cryptographic operation, however, is only used when the hash is computed inside the TOE.

6.2.2 FDP: User data protection

6.2.2.1 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: **O.PrivateKey and any working copies of ECC private key values**.

6.2.2.2 FDP_SDI.2: Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **[assignment: integrity failure of data stored in RAM, authentication and correspondence failures in imported blobs]** on all objects, based on the following attributes: **[assignment:**

- RAM
  - SECDED ECC value,
- Blobs
  - authenticity verification,
  - integrity check**]**.

Application note

This security attributes cannot be changed by default, queried, modified or deleted by the user.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall:

- **prevent use of altered data*;***
- **[assignment: zeroize the RAM,**
- **Indicate to the Cortex-M3 and Cortex-A7 that the TOE is in panic-mode]**

6.2.2.3 FDP_ACC.1: Subset access control

FDP_ACC.1.1 The TSF shall enforce the **Private Key Access Control SFP** on**:**

Subjects:	**S.User**
Objects:	**O.PrivateKey**
Operations:	**OP.KeyPair_Gen, OP.Signature, OP.EncDec**
	[assignment: OP.Key_derive].

6.2.2.4 FDP_ACF.1: Security attribute-based access control

FDP_ACF.1.1 The TSF shall enforce the **Private Key Access Control SFP** to objects based on the following:

Subjects: **S.User;**

Objects: **O.PrivateKey;**

Security attributes: [assignment:

- **blob type,**
- **authenticity verification of the request,**
- **integrity check of the request].**

Application note

Blob type is specified during key generation and identified during key import. It cannot be changed by default, queried, modified or deleted by the user when is under control of the TSF.

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **O.PrivateKey can only be accessed by S.User through operations involving private keys (OP.KeyPair_Gen, OP.Signature, OP.EncDec);**
- **[assignment: O.PrivateKey also can be accessed by S.User through operation involving private keys (OP.Key_derive)**
- **S.User can perform any of the operations defined in FDP_ACF.1.2 over O.PrivateKey by following the rules:**
  - **if OP.KeyPair_Gen: the blob type specified by S.User is valid for generating O.PrivateKey as is described in FCS_CKM.1/ECC**
  - **if OP.Signature, OP.EncDec and OP.Key_derive:**
    - **verification of authenticity and integrity of the request is positive**

      and

    - **the blob type specified by S.User is consistent to the blob containing O.PrivateKey].**

FDP_ACF.1.3 The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none.**

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the following additional rules:

- **No one shall be able to retrieve O.PrivateKey in a form allowing usage outside of the TOE;**
- **[assignment:**
  - **Failure of authenticity verification and/or integrity check of the request**
  - **if OP.Signature, OP.EncDec and OP.Key_derive, the blob type specified by S.User is not consistent to the blob containing O.PrivateKey].**

6.2.2.5 FDP_ACC.1 (Import_AE): Subset access control

FDP_ACC.1.1/Import_AE The TSF shall enforce the **PrivateKey Import PCK SFP** on

Subject: **S.User**
Operation: **OP.Import**

6.2.2.6 FDP_ACF.1 (Import_AE): Access control functions

FDP_ACF.1.1/Import_AE The TSF shall enforce the **PrivateKey Import PCK SFP** to objects based on the following:

Subject: **S.User**,

Object: **O.PrivateKey,**

Security attributes: [assignment:

- **blob type,**
- **authenticity verification of the request,**
- **integrity check of the request].**

Application note

Blob type is specified during key generation and identified during key import. It cannot be changed by default, queried, modified or deleted by the user when is under control of the TSF.

FDP_ACF.1.2/Import_AE The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **S.User is allowed to perform OP.Import, i.e. to import O.PrivateKey, only after successful verification according to FCS_COP.1/Import_Ver and successful decryption according to FCS_COP.1/Import_Dec);**
- **[assignment:**
  - **S.User can perform OP.Import over O.PrivateKey when:**
    - **verification of authenticity and integrity of the request is positive**

    and

    - **the blob type specified by S.User is consistent to the blob containing O.PrivateKey].**

FDP_ACF.1.3/Import_AE The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none.**

FDP_ACF.1.4/Import_AE The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[assignment:**

- **Failure of authenticity verification and/or integrity check of the request**
- **Blob type is not consistent to the blob used in OP.Import].**

6.2.2.7 FDP_ITC.1 (Import_AE): Import of user data without security attributes

FDP_ITC.1.1/Import_AE The TSF shall enforce the **PrivateKey Import PCK SFP** when importing ***O.PrivateKey*** ~~***user data***~~, controlled under the SFP, from outside of the TOE.

FDP_ITC.1.2/Import_AE The TSF shall ignore any security attributes associated with ***O.PrivateKey*** ~~***user data***~~ when imported from outside the TOE.

FDP_ITC.1.3/Import_AE The TSF shall enforce the following rules when importing ***O.PrivateKey*** ~~user data~~ controlled under the SFP from outside the TOE: **[assignment: no other rules]**.

6.2.3 FPT: Protection of the TSF

6.2.3.1 FPT_FLS.1: Failure with preservation of secure state

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:

- **Failing self-test according to FPT_TST.1;**
- **[assignment: Unexpected hardware behaviour (causing the TOE to enter panic mode)].**

Application Note

The secure state includes, but may not be restricted to, disabling access to the Secure Services. The secure state will be preserved until handled, which may require e.g., maintenance, service or repair of “hard” failures or only initialisation or resetting in case of “soft” failures.

6.2.3.2 FPT_PHP.3: Resistance to physical attack

FPT_PHP.3.1 The TSF shall resist **physical tampering** to the **all-TOE components implementing the TSF** by responding automatically such that the SFRs are always enforced.

Application Note

The TOE is not always powered and therefore not able to detect, react or notify that it has been subject to tampering. Nevertheless, its design characteristics make reverse-engineering and manipulations etc. more difficult. This is regarded as being an “automatic response” to tampering. Therefore, the security functional component Resistance to physical attack (FPT_PHP.3) has been selected. The TOE may also provide features to actively respond to a possible tampering attack which is also covered by FPT_PHP.3.

6.2.3.3 FPT_TST.1: TSF testing

FPT_TST.1.1 The TSF shall run a suite of self-tests **[selection: during initial start-up, periodically during normal operation, at the request of the authorised user, at the conditions [assignment: conditional tests defined in Table 16]]** to demonstrate the correct operation of the Secure Services **[selection: the TSF]**.

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of **[selection: TSF data]**.

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of **[selection: TSF]**.

Application Note

The TOE runs self-tests on start-up and periodically, at given time intervals. Two additional options are selected in FPT_TST.1.1 selection:

- Test at the request of the authorised users corresponds to a set of invokable APIs for running on-demand self-tests.

- Conditional tests are automatically executed each time a certain condition is satisfied.

Table 16 below summarizes the list of self-tests contemplated in FPT_TST.1.1 and the conditions under which they occur.

FPT_TST.1.1 selection	Self-tests
During initial start-up	• Firmware integrity test in ROM memory
Periodically during normal operation	• Firmware integrity test in ROM memory
At the request of the authorised user	• Cryptographic engine functions • DRBG • Firmware integrity test in ROM memory
Conditional tests	• Data integrity test in RAM memory made in every read operation

Table 16. FPT_TST.1.1 self-tests and triggering conditions

6.3 Security Assurance Requirements

The development and the evaluation of the TOE shall be done in accordance to the following security assurance package: **EAL4 + AVA_VAN.4, ALC_FLR.1**

The following table shows the assurance requirements by reference the individual components in **[CC31R5P3]**.

Assurance Class	Assurance Components
ASE: Security Target evaluation	ASE_CCL.1: Conformance claims ASE_ECD.1: Extended components definition ASE_INT.1: ST introduction ASE_TSS.1: TOE summary specification ASE_OBJ.2: Security objectives ASE_REQ.2: Derived security requirements ASE_SPD.1: Security problem definition
ALC: Life-cycle support	ALC_DEL.1: Delivery procedures ALC_LCD.1: Developer defined life-cycle model ALC_TAT.1: Well-defined development tools ALC_CMC.4: Production support, acceptance procedures and automation ALC_CMS.4: Problem tracking CM coverage ALC_DVS.1: Identification of security measures ALC_FLR.1: Basic flaw remediation
ADV: Development	ADV_ARC.1: Security architecture description ADV_FSP.4: Complete functional specification

Assurance Class	Assurance Components
	ADV_IMP.1: Implementation representation of the TSF ADV_TDS.3: Basic modular design
AGD: Guidance documents	AGD_OPE.1: Operational user guidance AGD_PRE.1: Preparative procedures
ATE: Tests	ATE_COV.2: Analysis of coverage ATE_DPT.1: Testing: basic design ATE_FUN.1: Functional testing ATE_IND.2: Independent testing – sample
AVA: Vulnerability assessment	AVA_VAN.4: Methodical vulnerability analysis

Table 17. Security Assurance Requirements

6.3.1 Refinements of the TOE Assurance Requirements

The following refinements shall support the comparability of evaluations according to the Protection Profile **[PP_V2XHSM].**

6.3.1.1 Refinements Regarding Preparative Procedures AGD_PRE.1

The following text states the requirements of the selected component AGD_PRE.1:

Developer action elements:

AGD_PRE.1.1D The developer shall provide the TOE including its preparative procedures.

Content and presentation elements:

AGD_PRE.1.1C The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_PRE.1.2C The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST. **Refinement:** ***The preparative procedures shall describe all necessary measures for integration with the VCS to guarantee the confidentiality, integrity and authenticity of the TOE assets according to OE. INTEGRATION.***

Evaluator action elements:

AGD_PRE.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AGD_PRE.1.2E The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared securely for operation.

6.4 Security Requirements Rationale

6.4.1 Security Functional Requirements Dependencies

Requirement	Direct explicit dependencies	Dependencies met by	Comment
FCS_CKM.1/ECC	[FCS_CKM.2 or FCS_COP.1] and FCS_CKM.4	FCS_COP.1/ECDSA FCS_COP.1/ECIES_ENC FCS_COP.1/ECIES_DEC FCS_CKM.4	
FCS_CKM.1/AES	[FCS_CKM.2 or FCS_COP.1] and FCS_CKM.4	FCS_COP.1/AES FCS_CKM.4	FCS_CKM.4 destruction method also covers symmetric keys
FCS_CKM.4	[FDP_ITC.1, or FDP_ITC.2, or FCS_CKM.1]	FCS_CKM.1/ECC FCS_CKM.1/AES FCS_CKM.5	
FCS_CKM.5	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	FCS_COP.1/ECDSA FCS_COP.1/ECIES_ENC FCS_COP.1/ECIES_DEC FCS_CKM.4	FCS_CKM.4 is defined in the base PP.
FCS_RNG.1	None	---	
FCS_COP.1/ECDSA	[FDP_ITC.1, or FDP_ITC.2, or FCS_CKM.1] FCS_CKM.4	FCS_CKM.1/ECC FCS_CKM.4	FCS_CKM.1/ECC is defined in the base PP. FCS_CKM.4 is defined in the base PP.
FCS_COP.1/ECIES_ENC	[FDP_ITC.1, or FDP_ITC.2, or FCS_CKM.1] FCS_CKM.4	FCS_CKM.1/ECC FCS_CKM.4	FCS_CKM.1/ECC is defined in the base PP. FCS_CKM.4 is defined in the base PP.
FCS_COP.1/ECIES_DEC	[FDP_ITC.1, or FDP_ITC.2, or FCS_CKM.1] FCS_CKM.4	FCS_CKM.1/ECC FCS_CKM.4	FCS_CKM.1/ECC is defined in the base PP FCS_CKM.4 is defined in the base PP.
FCS_COP.1/Import_Ver	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	FDP_ITC.1/Import_AE FCS_CKM.4	
FCS_COP.1/Import_Dec	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	FDP_ITC.1/Import_AE FCS_CKM.4	
FCS_COP.1/AES	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	FCS_CKM.1/AES FCS_CKM.4	
FDP_RIP.1	None	---	
FDP_SDI.2	None	---	
FDP_ACC.1	FDP_ACF.1	FDP_ACF.1	
FDP_ACF.1	FDP_ACC.1	FDP_ACC.1	

	FMT_MSA.3	Not met	See justifications below.
FDP_ACC.1/Import_AE	FDP_ACF.1	FDP_ACF.1/Import_AE	
FDP_ACF.1/Import_AE	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1/Import_AE Not met	See justifications below.
FDP_ITC.1/Import_AE	[FDP_ACC.1 or FDP_IFC.1] FMT_MSA.3	FDP_ACC.1/Import_AE Not met	See justifications below.
FPT_FLS.1	None	---	
FPT_PHP.3	None	---	
FPT_TST.1	None	---	

Table 18. SFR Dependencies

The rationale for SFR dependencies not met is as follows:

- FDP_ACF.1 dependency on FMT_MSA.3 is not applicable because the security attributes are defined from outside the TOE and imported in the TOE for the operations described in FDP_ACF.1. Therefore, no initialisation is done by the TOE.
- FDP_ACF.1/Import_AE and FDP_ITC.1/Import_AE dependencies on FMT_MSA.3 are not required because no initialisation is needed for import.
- FMT_MSA.1 and FMT_SMR.1 dependencies on FMT_MSA.3 are not met because the access control made over the ports depends on the physical port where the processor is attached. This is determined by the physical design of the TOE and there are no logical attributes which can be queried, modified or deleted. Regarding the roles, the TSF only considers one user and does not define user role.

6.4.2 Security Assurance Dependencies Analysis

The chosen evaluation assurance level EAL4 augmented by ALC_FLR.1 and AVA_VAN.4. Since all dependencies are met internally by the EAL package only the augmented assurance components dependencies are analysed.

Assurance Component	Dependencies	Met
ALC_FLR.1	None	Yes
AVA_VAN.4	ADV_ARC.1 Security architecture description	Yes
	ADV_FSP.4 Complete functional specification	Yes
	ADV_TDS.3 Basic modular design	Yes
	ADV_IMP.1 Implementation representation of the TSF	Yes
	AGD_OPE.1 Operational user guidance	Yes
	AGD_PRE.1 Preparative procedures	Yes
	ATE_DPT.1 Testing: basic design	Yes

Table 19. Security Assurance Dependencies Analysis

According to **Table 19**, all dependencies are met.

6.4.3 Security Functional Requirements Coverage

SFR/TOE Security Objective	OT.PRIVATEKEY_ACCESS	OT.SIGNATURE_GENERATION	OT.KEY_MANAGEMENT	OT.ECIES	OT.TOE_SELF-PROTECTION	OT.RNG	OT.VCS_DATA	OT.PRIVKEY_IMPORT_AE	OT.KEY_DERIVE
FCS_CKM.1/ECC		X							
FCS_CKM.1/AES		X							
FCS_CKM.4		X							
FCS_CKM.5									X
FCS_RNG.1		X	X			X			
FCS_COP.1/ECDSA		X							
FCS_COP.1/ECIES_ENC				X					
FCS_COP.1/ECIES_DEC				X					
FCS_COP.1/Import_Ver								X	
FCS_COP.1/Import_Dec								X	
FCS_COP.1/AES			X						
FDP_RIP.1			X						
FDP_SDI.2			X				X		
FDP_ACC.1	X								
FDP_ACF.1	X								
FDP_ACC.1/Import_AE								X	
FDP_ACF.1/Import_AE								X	
FDP_ITC.1/Import_AE								X	
FPT_FLS.1					X				
FTP_PHP.3					X		X		
FTP_TST.1					X				

Table 20. Security Functional Requirements Coverage

6.4.4 Security Functional Requirement Sufficiency

Objective	SFR	Rationale
OT.PRIVKEY_ACCESS	FDP_ACC.1 FDP_ACF.1 FCS_CKM.1/AES FCS_CKM.4 FCS_COP.1/Import_Ver FCS_COP.1/Import_Dec FCS_COP.1/AES	The TOE shall protect private key assets (FDP_ACC.1, FDP_ACF.1). Keys stored outside the TOE are protected (FCS_CKM.1/AES, FCS_CKM.4, FCS_COP.1/Import_Ver, FCS_COP.1/Import_Dec, FCS_COP.1/AES)
OT.SIGNATURE_GENERATION	FCS_RNG.1, FCS_COP.1/ECDSA	Signature generation is performed using ECDSA (FCS_RNG.1 and FCS_COP.1/ECDSA).
OT.KEY_MANAGEMENT	FCS_CKM.1/ECC FCS_CKM.1/AES FCS_CKM.4 FCS_RNG.1 FCS_COP.1/AES FDP_RIP.1 FDP_SDI.2	The TOE shall be able to generate ECC asymmetric key pairs (FCS_CKM.1/ECC) and symmetric key pair (FCS_CKM.1/AES) using RNG (FCS_RNG.1). The TOE shall be able to destroy key and key material (FCS_CKM.4, FDP_RIP.1). The TOE should protect the integrity of these keys (FCS_COP.1/AES) during the storage (FDP_SDI.2). The TOE protects the keys stored in storage external to the TOE in a blob structure (FCS_CKM.1/AES). Note: Confidentiality is covered by OT.PRIVKEY_ACCESS.
OT.ECIES	FCS_COP.1/ECIES_ENC FCS_COP.1/ECIES_DEC	The TOE shall be able to manage the ECIES operations (FCS_COP.1/ECIES_ENC, FCS_COP.1/ECIES_DEC) Note: Internal ECC key creation is covered by OT.KEY_MANAGEMENT.
OT.TOE_SELF-PROTECTION	FPT_FLS.1 FPT_PHP.3 FPT_TST.1	The TOE for its self-protection shall detect and react failures (FPT_TST.1) and preserve the secure state (FPT_FLS.1), as well as the resistance against tampering (FPT_PHP.3).
OT.RNG	FCS_RNG.1	The TOE shall implement secure RNG.
OT.VCS_DATA	FDP_SDI.2 FPT_PHP.3	The TOE shall guarantee the integrity of the stored data (FDP_SDI.2) and their confidentiality through resistance to tampering attacks (FPT_PHP.3).
OT.PRIVKEY_IMPORT_AE	FDP_ITC.1/Import_AE FDP_ACC.1/Import_AE FDP_ACF.1/Import_AE FCS_COP.1/Import_Ver FCS_COP.1/Import_Dec.	OT.PRIVKEY_IMPORT_AE is addressed by the implementation of FDP_ITC.1/Import_AE; the details of transfer protections are defined in FDP_ACC.1/Import_AE and FDP_ACF.1/Import_AE according to FCS_COP.1/Import_Ver and FCS_COP.1/Import_Dec.

OT.KEY_DERIVE	FCS_CKM.5	OT.KEY_DERIVE is addressed by FCS_CKM.5 which requires the implementation of derivation algorithm.

Table 21 Security Functional Requirements Sufficiency

6.4.5 Justification of the Chosen Evaluation Assurance Level

The assurance level EAL4 augmented with AVA_VAN.4 and ALC_FLR.1 has been chosen as appropriate for a Secure Hardware Module resisting threat agents possessing a Moderate attack potential.

7 TOE Summary Specification

In the following section is included a description of how the TOE satisfies all the SFRs that have been included in the corresponding section:

7.1 SF.RNG – Random number generation

This section covers the **Random Number Generation** functionality.

The TOE includes a DRBG in charge of generating random numbers for the cryptographic operations in scope, which include ECC private key generation (used for ECDSA-based signatures, ECIES and key derivation), symmetric key generation (AES), seeding of the cryptographic engine's countermeasures and RNG service for the VCS.

This Security Function is implemented by **FCS_RNG.1**.

7.2 SF.Key-Mgmt – V2X Key Management

This section covers the **V2X Key Management** functionality.

The V2X key management security function includes key generation, key secure import, key secure export and key destruction that is explained along this section.

The ECC keys used for signature generation (**FCS_COP.1/ECDSA**), ECIES decryption (**FCS_COP.1/ECIES_DEC**) and key derivation (**FCS_CKM.5**) are not permanently stored within the TOE. Instead, they are exported outside the TOE by adding encryption with specifically generated keys (**FCS_CKM.1/AES**, **FCS_COP.1/AES**).

The exported keys can be Eimported into the TOE by the appropriate means (**FCS_COP.1/Import_Dec**, **FCS_COP.1/Import_Ver**).

This procedure is used for importing and exporting keys that can be generated internally (**FCS_CKM.1/ECC**) or generated outside the TOE and imported (**FDP_ACC.1/Import_AE**, **FDP_ACF.1/Import_AE**).

The keys within the TOE are deallocated and destroyed when no longer required (**FDP_RIP.1**, **FCS_CKM.4**).

While performing TOE functions, the TOE monitors and detects any data modification (**FDP_SDI.2**).

7.3 SF.Crypto – Cryptographic operations

This section covers the **Digital Signature Generation, ECIES encryption/decryption** and **ECC key derivation** functionalities.

The TOE performs **digital signature generation** using externally or internally computed hash according to **[FIPS 186-4]** with NIST and Brainpool standard curves. This is used for signing V2X-related communication messages (**FCS_COP.1/ECDSA**).

The TOE implements **ECIES encryption and decryption** in order to encrypt and decrypt a VCS data encryption key used for secure communications through the V2X protocol (**FCS_COP.1/ECIES_ENC**, **FCS_COP.1/ECIES_DEC**).

According to ECIES scheme, the TOE implements HMAC-SHA operations to provide message authentication and integrity as well as support the key derivation for ECIES cryptographical operations. Therefore, the TOE implements these functions as specified in **[FIPS 180-4]**.

The access to the operations is controlled by physical security attributes defined by specific security policy (**FDP_ACC.1**, **FDP_ACF.1**).

7.4 SF.Self-Prot – Self-protection

This section covers the **Self-protection** functionality.

The TOE implements protection against physical attacks as follows:

- The hardware integration of the TOE in the SoC contributes to enforce self-protection of the TSF: as part of the SoC physical domain, the high integration of the TOE and the use of the same power domain as the rest of the modules in the SoC.
- Disabling of unused APIs to prevent access to functionality not intended to be used by the VCS.
- Furthermore, the TOE provides several secure coding practices to prevent the TOE against physical attacks.
- The TOE also implements protection against side-channel attacks.
- Finally, the TOE includes protection against fault injection.

This function is implemented by **FPT_PHP.3**.

The TOE also runs self-tests for verifying the integrity of the firmware during start-up. These tests are also executed periodically during normal operation of the TOE (**FPT_TST.1**).

Self-tests are also performed on demand for verifying the correct operation of the cryptographic engines, DRBG and the proper integrity of firmware (**FPT_TST.1**).

Finally, self-tests can be invoked when some conditions are met (**FPT_TST.1**).

The invocation of the self-tests is done by the user as follows:

- Power-up and periodic self-tests by calling the eHSM initialization API with the required configuration.
- On-demand whenever needed by calling the self-tests APIs.

Self-protection is also guaranteed by a set of protection routines which are executed in certain circumstances:

- The tamper detection routine is activated when tampering is raised in the TOE by the operational environment (**FPT_PHP.3**). If this condition is raised, the TOE will destroy all cryptographic secrets (**FCS_CKM.4**).
- The panic routine is activated when a self-test fails (**FPT_TST.1**) or in hardware failure (**FPT_FLS.1**) or after a tamper detection event (**FPT_PHP.3**). This panic routine performs keys zeroization (**FCS_CKM.4**), halting the Cortex-M0 processor and indicates the panic mode to the SaS.

Finally, exported user data (mainly user keys) to be stored outside the TOE are protected by encryption and authentication as is mentioned in section 7.2.

8 Acronyms

The following table shows the acronyms used in this document.

Acronym	Meaning
AES	Advanced Encryption Standard
API	Application Programming Interface
AT	Authorization Ticket, a.k.a. Pseudonym Certificate (PC)
CA	Certification Authority
CC	Common Criteria
C-ITS	Cooperative Intelligent Transport System
CPU	Central Processing Unit
DRBG	Deterministic Random Bit Generator
EAL	Evaluation Assurance Level
EC	Enrolment Credentials, a.k.a. Long-Term Certificate (LTC)
ECC	Elliptic Curve Cryptography or Error Correction Code, depending on the context
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
eHSM	Embedded Hardware Security Module
IC	Integrated Circuit
IT	Information Technology
ITS	Intelligent Transport System
IVN	In-Vehicle Network
KEK	Key Encryption Key;
MA	Member Authority
MAC	Message Authentication Code
OSP	Organisational Security Policies
OTP	One-Time-Programable Memory
RBG	Random bit generator
PP	Protection Profile
SaS	SoC Application Side
SoC	System on Chip
ST	Security Target
TOE	Target of Evaluation
TRNG	True Random Number Generator

Acronym	Meaning
TSF	TOE Security Functionality
TSFi	TSF Interface
V2X	Vehicle to Anything
VCS	Vehicle C-ITS Station
NVM	Non-volatile memory

Table 22. Acronyms

9 Glossary of Terms

Term	Meaning
Augmentation	Addition of one or more requirement(s) to a package
Evaluation Assurance Level	Set of assurance requirements drawn from CC Part 3, representing a point on the CC predefined assurance scale, that form an assurance package
Operational Environment	Environment in which the TOE is operated
Protection Profile	Implementation-independent statement of security needs for a TOE type
Security Target	Implementation-dependent statement of security needs for a specific identified TOE
Target Of Evaluation	Set of firmware, firmware and/or hardware possibly accompanied by guidance
SoC Application Side	The SoC Application side (SAS) is the other side of the eHSM boundary definition. Is the part of the SoC that is not the eHSM.

Table 23. Glossary of terms

10 Document References

The following table shows the documents referenced in this document.

Reference	Document
[CC31R5P1]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 1: Introduction and general model
[CC31R5P2]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 2: Security functional components
[CC31R5P3]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 3: Security assurance components
[CEM31R5P3]	Common Criteria Evaluation methodology, Version 3.1, Revision 5
[FIPS 180-4]	Secure Hash Standard, August 2015
[FIPS 186-4]	FIPS publication Digital Signature Standard (DSS), July 2013
[FIPS 198-1]	The Keyed-Hash Message Authentication Code (HMAC), July 2008
[FSP]	SECTON/CRATON2 Embedded V2X HSM CUT3.1 Functional Specification v1.4
[IEEE 1363a]	IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques, 2004
[IEEE 1609.2.1]	IEEE Std. 1609.2.1-2020: “Standard for Wireless Access in Vehicular Environments (WAVE) – Certificate Management Interfaces for End Entities”. Version December 2020.
[IEEE 1609.2]	IEEE Std 1609.2™ – 2016 including amendments IEEE Std 1609.2a™ – 2017 and IEEE Std 1609.2b™: “IEEE Standard for Wireless Access in Vehicular Environments – Security Services for Applications and Management Messages”. Version 2016 amended by 2017 and 2019
[MSSR]	JIL, Minimum Site Security Requirements. V3.0, February 2020
[OPE]	SECTON-CRATON2 Embedded V2X HSM CUT3.1 Operational Guidance v0.9
[PP_V2XHSM]	Protection Profile V2X Hardware Security Module, Version 1.0.1, Release 1.6.0, 30.11.2021
[PRE]	SECTON/CRATON2 Embedded V2X HSM CUT3.1 Preparational Guidance v1.3
[RFC 5639]	Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, March 2010
[SM-P001]	SM-P001 – Autotalks Development Security Manual v0.21
[SP800-133]	NIST Special Publication 800-133, Recommendation for Cryptographic Key Generation, December 2012
[SP800-38A]	Recommendation for Block Cipher Modes of Operations, Methods and Techniques, December 2001
[SP800-38B]	Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication

Reference	Document
[SP800-38C]	Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, May 2004
[SP800-90A]	NIST Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 2015
[SP800-90B]	NIST Special Publication 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generator, January 2018
[TS 102 941]	Intelligent Transport Systems (ITS); Security; Trust and Privacy Management., version 1.4.1
[TS 103 097]	Intelligent Transport Systems (ITS); Security; Security Header and Certificate Formats., version 1.3.1

Table 24. List of document references