Security Target for Raspberry Pi RP2350 in Secure mode for PSA Certified™ Level 2

Document number:	
Version:	Release
Release Number:	3
Author	Raspberry Pi and Arm
Date of Issue:	15/07/2025

Non-Confidential Proprietary Notice

Contents

1 About this document

1.1 Current Status and Anticipated Changes

Current Status: Released, version 2.0 REL 01

1.2 Release Information

The change history table lists the changes that have been made to this document.

Date	Version	Confidentiality	Change
18/11/2024	Beta 1	Released under NDA	Initial
29/11/2024	Beta 2	Released under NDA	Response to Juan's Comments
29/01/2025	Beta 3	Released under NDA	Response to Juan's Comments
30/01/2025	Beta 4	Released under NDA	Removed PP front Matter. Updated Change log
04/02/2025	Beta 5	Released under NDA	Update to A4 Fixed minor typos
12/02/2025	Release	Public	Changed release status
03/07/2025	Release 02	Public	Added reference to Raspberry PI integration Guide
15/07/2025	Release 03	Public	Corrected Pico SDK version required for A4.

1.3 References

This document refers to the following documents.

1.3.1 Normative references

Ref	Doc No	Author(s)	Title
[PSA-L1]	JSADEN001	JSA	PSA Certified Level 1 Questionnaire
[PSA-EM-L2]	JSADEN003	JSA	PSA Certified: Evaluation Methodology for PSA L2
[PSA-EM-L3]	JSADEN010	JSA	PSA Certified: Evaluation Methodology for PSA L3
[PSA-AM]	JSADEN004	JSA	PSA Certified Attack Methods
[PSA-PP-L2]	JSADEN002	JSA	PSA Certified Level 2 Lightweight Protection Profile
[PSA-PP-L3]	JSADEN009	JSA	PSA Certified Level 3 Lightweight Protection Profile
[SESIP-PP-L3]	JSADEN011	JSA	SESIP Profile for PSA Certified™ Level 3
[PSA-L2-COMP]	JSADEN017	JSA	SESIP Profile for PSA Certified™ RoT Component Level 2
[PSA-L3-COMP]	JSADEN018	JSA	SESIP Profile for PSA Certified™ RoT Component Level 3
[PSA-L4-iSE-SE]	JSADEN023	JSA	SESIP Profile for PSA Certified™ Level 4 iSE/SE and RoT Component
[SESIP]	GP_FST_070	GlobalPlatform	Security Evaluation Standard for IoT Platforms (SESIP) v1.2
[CEN-SESIP]	EN 17927	CEN/CENELEC	Security Evaluation Standard for IoT Platforms (SESIP) 2023
[CEM]	CCMB-2017-04-004	Common Criteria	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1, revision 5, April 2017.

1.3.2 Informative references

Ref	Doc No	Author(s)	Title
[GP-ROT]	GP_REQ_025	GlobalPlatform	Root of Trust Definitions and Requirements, Version 1.1, Public Release, June 2018
[PSA-SM]	JSADEN014	ARM	Platform Security Model
[PSA-SS]	IHI 0087	ARM	PSA Certified Secure Storage API, Version 1.0 or later
[SP-800-57]	SP 800-57 Part 1	NIST	Recommendation for Key Management: Part 1 – General, Rev. 5

1.4 Terms and Abbreviations

This document uses the following terms and abbreviations (see PSA-SM and PSA-L1).

Term	Meaning
Application	Used in this SESIP profile to refer to the components which are out of the scope of the evaluation.
Application Root of Trust Service(s)	Application specific security service(s) that are not defined by PSA. Such services execute in the Secure Processing Environment and are required to be in Secure Partitions.
Application Specific Software	Software that provides the functionality required of the specific device. This software runs in the Non-Secure Processing Environment, making use of the System Software, Application RoT Services and PSA-RoT Services.
Critical Security Parameter	Secret information, with integrity and confidentiality requirements, used to maintain device security, such as authentication data (passwords, PIN, certificates), secret cryptographic keys, etc..
Evaluation Laboratory	Laboratory or facility that performs the assessment of products submitted for PSA Certified. The list of evaluation laboratories participating to PSA Certified can be found on www.psacertified.org
Hardware Unique Key (HUK)	Secret and unique to the device symmetric key that must not be accessible outside the PSA Root of Trust. It is a Critical Security Parameter.
Host Platform	The entity which when used in composition with a certified PSA Level 2 RoT Component [PSA-L2-COMP], a certified PSA Level 3 RoT Component [PSA-L3-COMP] or a certified PSA Level 4 RoT Component [PSA-L4-iSE-SE] form the scope of the certification covered in this profile.
Initial Attestation Key (IAK)	A PSA-RoT secret private key from an asymmetric key-pair used to sign attestation reports, thus ensuring that the report is bound to a unique PSA- RoT (and so device) instance.
Non-secure Processing Environment (NSPE)	The processing environment that hosts the non-secure System Software and Application Specific Software. PSA requires the NSPE to be isolated from the SPE. Isolation between partitions within the NSPE is not required by PSA though is encouraged where supported.
Partition	The logical boundary of a software entity with intended interaction only via defined interfaces, but not necessarily isolated from software in other partitions. Note that both the NSPE and SPE may host partitions.
Platform	Used in this SESIP Profile to refer to the components which are in the scope of the evaluation.
PSA	Platform Security Architecture
PSA Certification Body	The entity that receives applications for PSA security certification, issues the certificates, maintains the security certification scheme, and ensures consistency across all the evaluation laboratories.
PSA Functional APIs	PSA defined Application Programming Interfaces on which security services can be built. APIs defined so far include Crypto, Secure Storage and Attestation.
PSA Functional API Certification	Functional certification confirms that the device implements the PSA Functional APIs correctly by passing the PSA Functional certification test suites.

Term	Meaning
PSA Root of Trust (PSA-RoT)	The PSA defined combination of the Immutable Platform Root of Trust and the Updateable Platform Root of Trust and is considered to be the most trusted security component on the device. See [PSA-SM].
Immutable Platform Root of Trust	The minimal set of hardware, firmware and data of the PSA-RoT, which is inherently trusted because it cannot be modified following manufacture. There is no software at a deeper level that can verify that it as authentic and unmodified.
Updateable Platform Root of Trust	The firmware, software and data of the PSA-RoT that can be securely updated following manufacture.
Platform Root of Trust Service(s)	PSA defined security services for use by PSA-RoT, Application RoT Service(s) and by the NSPE. Executes in the Secure Processing Environment and may use Trusted Subsystems. This includes the services offered by the PSA Functional APIs.
SESIP Profile	Document providing a common set of functionalities for similar products
Secure Partition	A Partition in the Secure Processing Environment.
Secure Processing Environment Partition Management	Management of the execution of software in Secure Partitions. Typical implementations will provide scheduling and inter partition communication mechanisms. Implementations may also enforce isolation between the managed Secure Partitions.
Secure Processing Environment (SPE)	The processing environment that hosts the PSA-RoT, the PSA-RoT Services, and any Application RoT Service(s).
Secure Boot	The process of verifying and validating the integrity and authenticity of updateable firmware and software components as a pre-requisite to their execution. This must apply to all the firmware and software in the SPE. It should also apply to the first NSPE image loaded, which may extend the NSPE secure boot chain further.
Security Target (ST)	Document providing an implementation-dependent statement of security of a specific identified platform.
System Software	NSPE software that may comprise an Operating System or some run-time executive, together with any middleware, standard stacks and libraries, chip specific device drivers, etc., but not the application specific software.
TOE	Target of Evaluation. In this SESIP Profile it is a synonym for Platform.
Trusted subsystem	A security subsystem that the PSA-RoT relies on for protection of its assets, or that implement some of its services.

2 Introduction

This SESIP profile proposes a mapping between the security functionality defined in the PSA L2 Protection Profile [PSA-PP-L2] and the SFRs (Security Functional Requirements) listed in the SESIP catalogue [SESIP]. This profile also includes some optional SFRs aiming to cover most of the platform use cases.

The effort for performing the AVA_VAN.2 activities of a standard implementation of a PSA-RoT is **25 person-days**. It is assumed for this workload that:

- the source code for the components in scope of the platform (see Sections 2.4.2 and 2.4.3, hardware design is not required). This shall include drivers for Trusted Subsystems if used;
- no additional SFRs are added in the Profile;
- evaluation activities are not re-used;
- the SFRs “https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/services/secure_partition_manager.html
- Cryptographic Operation” and “a true hardware random source, which is then passed to a SHA256 DRBG compliant with NIST 800-90B and AIS-31 provided by TF-M.

The hardware TRNG is described in [RPI] 12.12.

- Cryptographic Key Generation” include one cryptographic algorithm;
- the platform does not rely on a certified trusted subsystem or certified PSA Certified RoT Component (see Sections 2.2 Platform Reference and 2.4.5 Required hardware/Software/Firmware).

2.1 SESIP Profile Reference

Reference	Value
PP Name	SESIP Profile for PSA Certified Level 2
PP Version	V2.0 Rel 01
Assurance Claim	SESIP Assurance Level 2 (SESIP2)
SESIP Standard	GP-SESIP
Optional and additional SFRs	Secure Data Serialization

Table 1: SESIP Profile Reference

2.2 Platform Reference

The platform is uniquely identified by its chip (hardware) reference and its PSA defined Root of Trust (software) reference as described below. The developer declares that only the evaluated and successfully certified products identify in this way.

Reference	**Value**	
Platform Name	Raspberry Pi RP2350 in Secure mode	
Platform Version	Raspberry Pi RP2350 – A4 with Build ROM version A4 Pico SDK v2.1.1 TF-M 2.1.1 LTS TF-M 2.1.1 includes: MCU Boot v2.1.0. Mbed TLS v3.6.2	
Platform Identification	Chip name and version	Pi RP2350 – A4 with Build ROM version A4
	PSA-RoT name and version	TF-M 2.1.1 LTS
Platform Type	System on Chip	
Trusted Subsystem Identification	None	
Trusted Sub-system Certification	N/A	

Table 2: Platform Reference

2.3 Included Guidance Documents

The following documents are included with the platform:

Reference	Name	Version
[TFM]	Trusted Firmware-M Documentation	https://tf-m-reference.trustedfirmware.org/index.html https://trustedfirmware-m.readthedocs.io Version 2.1.1 The documentation source can be found at: https://git.trustedfirmware.org/plugins/gitiles/TF-M/trusted-firmware-m.git/+/refs/tags/TF-Mv2.1.1/docs/
[RPD]	RP2350 datasheet	https://datasheets.raspberrypi.com/rp2350/rp2350-datasheet.pdf Build 5e790a3-clean

Table 3: Guidance Documents

2.4 Platform Functional Overview and Description

2.4.1 Platform Type

The hardware is a *System-on-Chip.*

2.4.2 Physical Scope

The platform has four cores, two of which are disabled.

In the evaluated configuration, Core 0 and Core 1 are both Arm M33 cores. The two Hazard 3 cores are disabled.

Both cores include internal hardware isolation.

However, the SPE is restricted to Core 0, Core 1 is restricted to non-secure operation.

The hardware is in the scope of the security evaluation as it provides security features, such as immutable storage, isolation features, cryptographic functionality and key management, which are essential for ensuring the security of the implementation.

ToE consist of an Arm Cortex-M33 CPU with security extensions, which allows the CPU to operate in a secure or non-secure mode. The secure mode implements the SPE, while the non-secure mode implements the NSPE. The SoC provides the hardware support for isolation of secure platforms.

The platform also includes TF-M LTS firmware which provides the Platform Root of Trust. TF-M provides the PSA Storage Service, PSA Cryptographic Service and PSA Initial Attestation Service. TFM also configures the isolation between the PRoT and any ARoT partitions.

In the evaluated configuration no ARoT partitions are configured.

2.4.3 Logical Scope

The scope for a SESIP Security evaluation, or Target of Evaluation (TOE), according to this profile is the combination of the trusted hardware and firmware components implementing a PSA-RoT with the Security Functional Requirements stated in this document, see Figure 1.

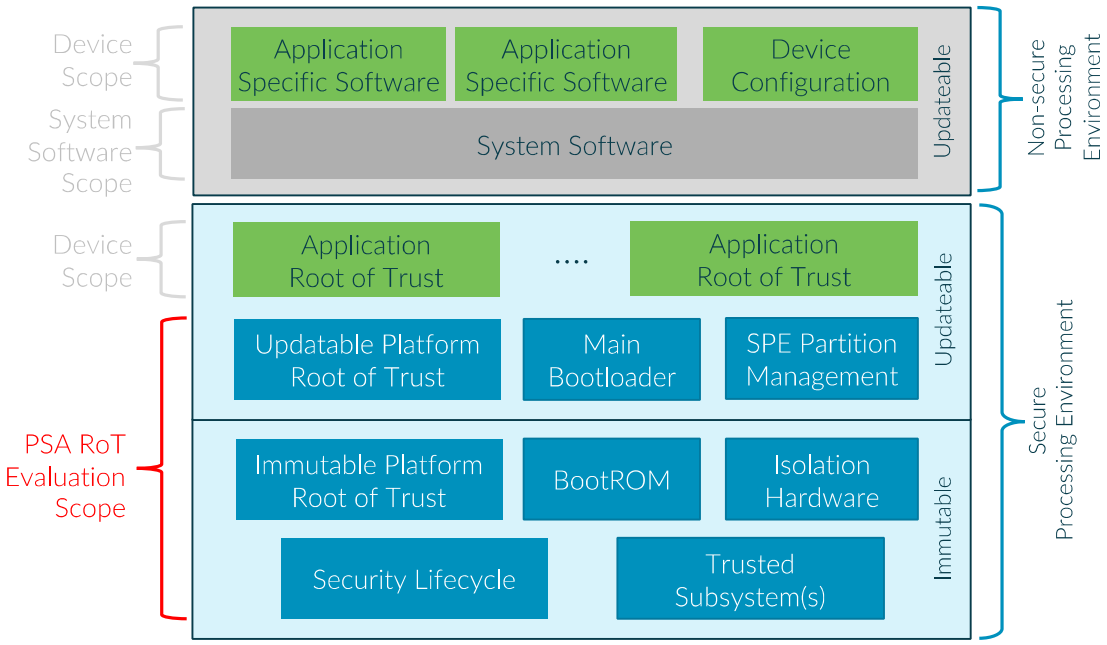

Figure 1: Scope of PSA Certified Level 2

The Chip security evaluation scope includes the following Secure Processing Environment PSA-RoT elements, as described in [PSA-SM]:

- Immutable Platform Root of Trust, for example, the Boot ROM, any root parameters, the NSPE/SPE isolation hardware, and any hardware-based security lifecycle management and enforcement.
- Updateable Platform Root of Trust, for example, a main bootloader, the code that implements the SPE Partition Management function, the code that implements the PSA defined services such as attestation, secure storage, and cryptography.
- Any Trusted subsystems that the host processor relies on for protection of its assets, or that implement some of its services.

The Platform scope hardware may be a System-on-Chip or a System-in-Package, possibly supported by board level trusted subsystem components, for example, a Secure Element or Subscriber Identification Module.

The Platform is a system-on-Chip, supported by external untrusted flash.

The software scope is an immutable bootloader (NSIB) as part of the immutable Platform Root of Trust, the updateable platform root of trust running from the main bootloader based on MCUboot, and an updatable system firmware based on Trusted Firmware-M running in the SPE. The secure firmware provides cryptographic functionality, secure storage and attestation APIs that are in the logical scope of evaluation.

2.4.4 Usage and Major Security Features

This profile considers the following features for the purpose of PSA Level 2 security evaluation:

- A Secure Processing Environment (SPE) isolated by hardware mechanisms to protect critical services and related assets from the Non-Secure Processing Environment.
- A Secure Boot process to verify integrity and authenticity of executable code in a chain of trust starting from the Boot ROM. Related certificates are protected in integrity by hardware mechanisms.
- Support for Secure Storage, to protect in integrity and confidentiality sensitive assets for the SPE and related applications. These assets include at least the Hardware Unique Key (HUK), the PSA-RoT Public Key (ROTPK), the Initial Attestation Key (IAK), and the unique instance ID.
- A Security Lifecycle for the SPE, to protect the lifecycle state for the device and enforce the transition rules between states.
- Cryptographic functions services for SPE and SPE applications.
- Support for an attestation method, for example Entity Attestation Token (according to IETF specification).

no additional information

2.4.5 Required Hardware/Software/Firmware

An external flash chip is required for non-volatile storage.

The data stored is encrypted and authenticated within the Secure Processing Environment.

Therefore, the flash is a Non-TOE component.

The external device just provides a storage area.

3 Security Objectives for the operational environment

For the platform to fulfil its security requirements, the operational environment (technical or procedural) must fulfil the following objectives.

ID	Description	Reference
KEY_MANAGEMENT	Cryptographic keys and certificates outside of the platform are subject to secure key management procedures.	[RPD] Section 3.1
TRUSTED_USERS	Actors in charge of platform management, for instance for signature of firmware update, are trusted.	[RPD] Section 3.2
UNIQUE_ID	The integrity and uniqueness of the unique identification of the platform must be provided by the platform user during the personalization stage.	[RPD] Section 3.3

Table 4: Security Objectives for the Operational Environment

3.1 Key Management

The firmware is supplied as source code that must be compiled and signed before the Chip is used in a secure product.

Those responsible for building the software must take adequate precautions to protect the Private and Secret keys.

It is recommended that the these are stored in a certified HSM.

3.2 Trusted Users

It is assumed that the firmware is built using the versions specified with the standard configuration for the Raspberry Pi 2350 platform.

In particular, no changes to the configuration settings that control security.

These settings are listed in the Appendix.

3.3 Unique ID

The security relies on each device having an unguessable HUK. Ideally, these would be generated by a cryptographically secure RNG and injected into the device during manufacture at a secure facility.

4 Security Requirements and Implementation

4.1 Security Assurance Requirements

The claimed assurance requirements package is **SESIP2** as described in Section 5.1.

4.1.1 Flaw Reporting Procedure (ALC_FLR.2)

In accordance with the requirement for a flaw reporting procedure (ALC_FLR.2), including a process to report flaw and generate any needed update and distribute it, the developers have defined the following procedures:

As the TOE contains hardware from Raspberry Pi and firmware from TrustedFirware.org there are two possible points of contact.

However, the two Security teams are in regular contact. If an issue is raised on one forum that requires a solution by the other team, the issue will be passed from one to the other.

Flaws in hardware, initial boot ROM and Pico SDK

Any user who has a security concerns or a disclosure to make, is encouraged to contact the security team via security@raspberrypi.com

Flaws in firmware (TF-M, Mbed TLS and MCUBoot)

The firmware is open-source software. Non-security issues can be reported on the mailing list or on GitHub.

However, there is a specific process for security critical incidents, this is described here:

https://trusted-firmware-docs.readthedocs.io/en/latest/security_center/incident_handling_process.html

4.2 Base PP Security Functional Requirements

4.2.1 Verification of Platform Identity

The platform provides a unique identification of the platform, including all its parts and their versions.

Rationale

Attestation is provided through the TF-M initial Attestation Service.

The attestation token includes the hardware version, the version of TF-M and a list of software components.

For more information see:

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/integration_guide/services/tfm_attestation_integration_guide.html

4.2.2 Verification of Platform Instance Identity

The platform provides a unique identification of that specific instantiation of the platform, including all its parts.

Rationale

The Instance ID is included in the signed token returned by the TF-M Initial Attestation Service.

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/integration_guide/services/tfm_attestation_integration_guide.html

4.2.3 Attestation of Platform Genuineness

The platform provides an attestation of the “Verification of Platform Identity” and “Verification of Platform Instance Identity”, in a way that ensures that the platform cannot be cloned or changed without detection.

Rationale

The platform provides an attestation of the “Verification of Platform Identity” and “Verification of Platform Instance Identity”, through the TF-M Initial Attestation Service.

The attestation token includes details of the platform. Only the instance containing the correct private key can sign a new token.

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/integration_guide/services/tfm_attestation_integration_guide.html

4.2.4 Secure Initialization of Platform

The platform ensures its integrity and authenticity during platform initialization. If the platform integrity or authenticity cannot be ensured, the platform will go to **a state where no other operation except optionally Secure Update of Platform can be performed**.

Rationale

On boot, the hardware logic of the SoC reads the state of the SECURE_BOOT_ENABLE critical OTP flag.

In the evaluated configuration, this flag MUST be set.

This forces the platform to use the Arm cores. And turns on the secure boot sequence, which is provided by the Immutable RoT (Boot ROM).

The image is hashed – using SHA256 - and the signature is verified using the public key provided. The hardware verifies that the SHA-256 hash of the public key matches the value stored in OTP and that the hash in the signature matches the hash calculated by the BootROM.

The sequence is described in detail Section 10.1.1 of [RPD]

The second stage boot is implemented by TF-M is based on MCUBoot.

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/booting/tfm_secure_boot.html

4.2.5 Attestation of Platform State

The platform provides an attestation of the state of the platform, such that it can be determined that the platform is in a known state.

Rationale

The attestation token provided by the initial attestation service includes details of the platform slate.

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/integration_guide/services/tfm_attestation_integration_guide.html

4.2.6 Secure Update of Platform

The platform can be updated to a newer version in the field such that the **confidentiality,** integrity and authenticity of the platform is maintained.

Rationale

The platform includes a secure firmware update service.

In the evaluated configuration, the device is booted in secure mode, therefore the firmware must be signed with the private key corresponding to the public key whose hash is stored in fuse.

The PSA Secure Firmware Update support encrypted and signed images. Images are loaded into a staging area and on a reboot are verified and if valid then marked as the new version.

In order to use encrypted images, the source code must be compiled with `MCU_ENC_IMAGES = TRUE`

The Raspberry Pi supports rollback prevention, the version of the candidate image is compared to the version number stored in fuse and if lower than the stored value, the image is rejected.

See

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/services/tfm_fwu_service.html

and Chapter 5 of the [RPD]

4.2.7 Software Attacker Resistance: Isolation of Platform (between SPE and NSPE)

The platform provides isolation between the application and itself, such that an attacker able to run code as an application on the platform cannot compromise the other functional requirements.

Rationale

The Arm M33 is TrustZone enabled. The SPE runs in the Secure world of CPU 0.

The NSPE runs in the Non-Secure world of CPU0 and on CPU 1.

The TrustZone mechanism ensures the code running in the Non-Secure partitions cannot access memory allocated to the secure partitions.

See section 10.1.3 of [RPD]

And

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/services/secure_partition_manager.html

4.2.8 Software Attacker Resistance: Isolation of Platform (between PSA-RoT and Application Root of Trust Services)

The platform provides isolation between the application and itself, such that an attacker able to run code as an application on the platform cannot compromise the other functional requirements.

Rationale

The platform uses TF-M to provide partitions for PRoT and any ARoT functionality.

Each partition has its own MPU region.

The RP 2350 provides 8 secure MPU regions. Therefore, the image can have up to seven ARoT partitions.

And attempt to access memory assigned to a different partition cause a fault.

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/services/secure_partition_manager.html

4.2.9 Cryptographic Operation

The platform provides Operations in Table 5 functionality with algorithms in Table 5 as specified in specifications in Table 5 for key lengths described in Table 5 and modes described in Table 5.

Rationale

Algorithm	Operations	Specification	Key lengths	Modes
AES	Encryption/Decryption Authenticated encryption with additional data	NIST FIPS 197 (AES) NIST SP800-38A (ECB, CBC, CFB, OFB, CTR) NIST SP800-38C (CCM) NIST SP800-38D (GCM)	128, 192 256	ECB, CTR CBC CFB OFB CCM GCM
ChaCha20-Poly1305	Authenticated encryption with additional data	RFC7539	256	
RSA	Encryption Decryption Signature generation Signature verification	PKCS#1	1024 2048 3072	PSS OAEP
ECDSA	Signature generation Signature verification	NIST FIPS 186-4 SEC 2	secp192r1 secp224r1 secp256r1 secp512r1 secp192k1 secp224k1 secp256k1	ECDSA
ECDH	Key agreement	NIST SP 800-56A rev. 2	secp192r1 secp224r1 secp256r1 secp512r1 secp192k1 secp224k1 secp256k1	ECDH
EdDSA	Signature generation Signature verification	Ed25519	Curve25519	EdDSA
x25519	Key agreement	x25519	Curve25519	x25519
SHA	Secure hashing Keyed hashing for HMAC	NIST FIPS 180-3	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	SHA
HMAC	Message authentication	RFC2104	HMAC-SHA1 HMAC-SHA-224 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	HMAC
HKDF	Key derivation	RFC5869 NIST SP800-56C Rev 2		HKDF

AES-CMAC	Message authentication and key derivation	NIST SP800-38B NIST SP800-108r1	128 192 256	

Table 5: Cryptographic Operations

4.2.10 Cryptographic Random Number Generation

The platform provides a way based on *a true hardware random source* to generate random numbers to as specified in *NIST 800-90B and AIS-31*

Rationale

The platform provides a true hardware random source, which is then passed to a SHA256 DRBG compliant with NIST 800-90B and AIS-31 provided by TF-M.

The hardware TRNG is described in [RPI] 12.12.

4.2.11 Cryptographic Key Generation

The platform provides a way to generate cryptographic keys for use in cryptographic algorithms in Table 6 as specified in specifications in Table 6 for key lengths described in Table 6.

Rationale

The platform uses the services provided by TF-M to generate keys for the application.

The Platform supports the following key types and sizes:

ID	Algorithm	Specification	Key lengths
RSA	RSA OAEP, RSA PSS, RSA PKCS#1	PKCS#1 v1.5/2.1	1024, 2048, 3072
ECC	ECDH ECDSA	SEC 2 FIPS 186-4 Ed25519 x25519	According to curve type
Symmetric keys	AES, ChaCha-Poly	NIST FIPS 197 NIST SP800-38C NIST SP800-38D ChaCha/Poly1305	AES: 128, 192, 256 ChaCha-Poly: 256

Table 6: Cryptographic Key Generation

4.2.12 Cryptographic KeyStore

The platform provides a way to store *cryptographic keys* such that not even the application can compromise the *authenticity, integrity, confidentiality* of this data. This data can be used for the cryptographic operations *all operations*.

Rationale

The application uses the PSA Cryptographic Services provided by TF-M.

These services use the ITS service to store all cryptographic keys.

The ITS ensures that objects stored are only returned to the partition that stored them.

All keys are generated by the PSA Crypto Service – and so only this services can see the value of the key.

All other services access keys by a handle that is not related to the key value.

On the RP2350, the ITS is provided by External Trusted Secure Storage.

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/services/tfm_its_service.html#encryption-in-its

The store is located in external flash, and is encrypted using AES 256 GCM, using a key derived from the Hardware Unique key, stored in fuse to ensure the confidentiality and integrity of the data.

All contents are encrypted within the secure partition of the Raspberry Pi chip, the flash device is considered a non-TOE component.

To ensure the flash contents are encrypted, the code must be compiled with the flag.:

```
ITS_ENCRYPTION =ON
```

4.3 Optional Security Functional Requirements

4.3.1 Secure Data Serialization

The platform ensures that all data stored outside the direct control of the platform, except for none, is protected such that the **authenticity, integrity, confidentiality** is ensured.

Rationale

The Raspberry Pi uses external flash to provide Secure Trusted Storage.

The store is located in external flash.

To ensure the confidentiality and integrity of the data the storage service encrypts all the data to be written to flash within the secure partition of the Raspberry Pi chip, using AES 256 GCM, using a key derived from the Hardware Unique key, stored in fuse.

As only ciphertexts leave the secure partition, the flash device provides no security and is considered a non-TOE component.

This is described in:

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/services/tfm_its_service.html#encryption-in-its

https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/design_docs/services/ps_key_management.html

To ensure the flash contents are encrypted, the code must be compiled with the flag.:

```
ITS_ENCRYPTION =ON
```

5 Mapping and Sufficiency Rationales

5.1 Assurance

The assurance activities defined in [PSA-EM-L2] fulfil the SESIP2 activities. In particular, the required source code review, vulnerability analysis and testing to an equivalent of 25 person-days of the [PSA-EM-L2] is applicable.

Assurance Class	Assurance Family	Covered by
ASE: Security Target evaluation	ASE_INT.1 ST Introduction	Section “Introduction” and title page of the Security Target
	Rationale:	
	ASE_OBJ.1 Security requirements for the operational environment	Section “Security Objectives for the Operational Environment” of the Security Target
	Rationale:	
	ASE_REQ.3 Listed Security requirements	Section “Security Requirements and Implementation” of the Security Target
	Rationale:	
	ASE_TSS.1 TOE Summary Specification	Section “Security Requirements and Implementation” of the Security Target
	Rationale:	
ADV: Development	ADV_FSP.4 Complete functional specification	The developer reference [RPM] and [TFM]
	Rationale:	
AGD: Guidance documents	AGD_OPE.1 Operational user guidance	https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/getting_started/
	Rationale:	
	AGD_PRE.1 Preparative procedures	https://trustedfirmware-m.readthedocs.io/en/tf-mv2.1.1/getting_started/
	Rationale:	
ALC: Life-cycle support	ALC_FLR.2 Flaw reporting procedures	ALC_FLR section in the Security Target and description of which developer evidence is used to meet this requirement
	Rationale:	
ATE: Tests	ATE_IND.1 Independent testing: conformance	Vulnerability and testing carried out by the laboratory
	Rationale:	
AVA: Vulnerability Assessment	AVA_VAN.3 Focused vulnerability analysis	Vulnerability and testing carried out by the laboratory
	Rationale:	

Table 7: Assurance Mapping and Sufficiency Rationales

6 Appendix: Build options

The TF-M firmware is supplied as source code and must be built and signed before loading

To ensure that TF-M offers the security features required to comply with this ST, the following options must be specified.

Flag	Value
BL2_TRAILER_SIZE	0x800
TFM_NS_MAILBOX_API	OFF
TFM_PARTITION_NS_AGENT_MAILBOX	OFF
TFM_NS_CUSTOM_API	OFF
CONFIG_TFM_USE_TRUSTZONE	ON
MCUBOOT_USE_PSA_CRYPTO	ON
MCUBOOT_SIGNATURE_TYPE	"EC-P256"
MCUBOOT_HW_KEY	OFF
MCUBOOT_BUILTIN_KEY	ON
PROVISIONING_CODE_PADDED_SIZE	"0x2000"
PROVISIONING_VALUES_PADDED_SIZE	"0x400"
PROVISIONING_DATA_PADDED_SIZE	"0x400"
TFM_MBEDCRYPTO_PLATFORM_EXTRA_CONFIG_PATH	<*TF-M root dir*> /platform/ext/target/rpi/rp2350/ mbedtls_extra_config.h
PLATFORM_DEFAULT_PROV_LINKER_SCRIPT	OFF
ITS_ENCRYPTION	ON
PLATFORM_DEFAULT_NV_SEED	OFF
PLATFORM_DEFAULT_OTP	OFF
PLATFORM_DEFAULT_NV_COUNTERS	OFF
PLATFORM_DEFAULT_CRYPTO_KEYS	OFF
PS_NS_NV_COUNTER_IN_ITS	ON
TFM_RP2350_MANIFEST_LIST	<*TF-M root dir*> /platform/ext/target/rpi/rp2350/manifest/ tfm_manifest_list.yaml
TFM_MANIFEST_LIST	<*TF-M root dir*>/ platform/ext/target/rpi/rp2350/manifest/ tfm_manifest_list.yaml

Table 8: Build options

Additional flags required for the build:

Flag	Value
TFM_PLATFORM	rpi/rp2350
TFM_TOOLCHAIN_FILE	<*TF-M source dir*>/toolchain_GNUARM.cmake
CONFIG_TFM_SOURCE_PATH	<*TF-M source dir*>\
TFM_PROFILE	profile_medium

Table 9: Additional build flags

OPTIONAL, Example additional flags for provisioning bundle, these flags tell the build to generate random HUK, IAK, RSA keypair:

Flag	Value
TFM_DUMMY_PROVISIONING	OFF
PLATFORM_DEFAULT_PROVISIONING	OFF
MCUBOOT_GENERATE_SIGNING_KEYPAIR	ON

Table 10: Additional flags to generate random HUK, IAK and RSA keypair

Additional flags required for confidentiality of code updates:

Flag	Value
MCUBOOT_ENC_IMAGES	TRUE

Table 11: Additional flag to ensure confidentiality of code updates

To ensure that the OTP bit disabling debug mode is set use the command:

```
picotool otp set OTP_DATA_CRIT1.DEBUG_DISABLE 1
```