RealSecuriTEE

SESIP Security Target for PSA Certified Level 2

Rev. 1.2
3rd Mar. 2026

Realtek Semiconductor Corp.

No. 2, Innovation Road II, Hsinchu Science Park, Hsinchu 300, Taiwan

Tel.: +886-3-578-0211. Fax: +886-3-577-6047

www.realtek.com

COPYRIGHT

©2025 Realtek Semiconductor Corp. All rights reserved. No part of this document may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form or by any means without the written permission of Realtek Semiconductor Corp.

DISCLAIMER

Realtek provides this document ‘as is’, without warranty of any kind. Realtek may make improvements and/or changes in this document or in the product described in this document at any time. This document could include technical inaccuracies or typographical errors.

TRADEMARKS

Realtek is a trademark of Realtek Semiconductor Corporation. Other names mentioned in this document are trademarks/registered trademarks of their respective owners.

LICENSE

USING THIS DOCUMENT

This document is intended for the software engineer’s reference and provides detailed programming information.

Though every effort has been made to ensure that this document is current and accurate, more information may have become available subsequent to the production of this guide.

ELECTROSTATIC DISCHARGE (ESD) WARNING

This product can be damaged by Electrostatic Discharge (ESD). When handling, care must be taken.

Damage due to inappropriate handling is not covered by warranty.

Do not open the protective conductive packaging until you have read the following, and are at an approved anti-static workstation.

- Use an approved anti-static mat to cover your work surface
- Use a conductive wrist strap attached to a good earth ground
- Always discharge yourself by touching a grounded bare metal surface or approved anti-static mat before picking up an ESD-sensitive electronic component
- If working on a prototyping board, use a soldering iron or station that is marked as ESD-safe

Always disconnect the microcontroller from the prototyping board when it is being worked on.

REVISION HISTORY

Revision	Release Date	Summary
1.0	2025/11/20	First release.
1.1	2026/02/06	Second release.
1.2	2026/03/03	Third release

Content

Tables

Figures

1. Introduction

The Security Target describes the Platform (in this chapter) and the exact security properties of the Platform that are evaluated against “GlobalPlatform Technology Security Evaluation Standard for IoT Platforms (SESIP), Version 1.2, GP_FST_070” (in chapter “Security requirements and implementation”) that a potential consumer can rely upon the product upholding if they fulfil the objectives for the environment (in chapter “Security Objectives for the operational environment”).

1.1. ST Reference

RealSecuriTEE, SESIP Security Target, Revision 1.2, Realtek Semiconductors, 3 Mar 2026.

1.2. SESIP Profile Reference

Reference	**Value**
PP Name	SESIP Profile for PSA Certified™ Level 2
PP Version	2.0 REL 01
Assurance Claim	SESIP Assurance Level 2 (SESIP 2)
Optional and additional SFRs	Secure Debugging Secure Encrypted Storage Reliable Index

Table 1 SESIP Profile Reference

1.3. Platform Reference

Reference	**Value**	
Platform Name	RealSecuriTEE	
Platform Version	V1.0	
Platform Identification	Chip name and version	Lalu platform security module V1.1
	PSA-RoT name and version	Siren trusted firmware V1.0
Platform type	Security Soft IP	

Table 2 Platform Reference

1.4. Terms and Abbreviations

Term	Meaning
CPU	Central Processing Unit
Lalu	Lalu Platform Secure Module
Siren	Siren Trusted Firmware
System	System Platform
ROM	The first stage of secure boot code runs in ROM
BL2 FW	The second stage of secure boot code runs in RAM

1.5. Included Guidance Documents

Reference	Name	Version
[FW_Manual]	RealSecuriTEE Firmware Reference Manual	V1.3 Date: 2025-12-11
[RUDG_GUID]	Realtek Vulnerability Disclosure Guidelines	V1.0 Date: 2025-03-21
[RRDP_GUID]	Realtek Responsible Disclosure Policy	V1.0 Date: 2025-03-21
[RPVHP_GUID]	Realtek Product Vulnerability Handling Process	V1.0 Date: 2025-03-21
[HW_INTRO]	RealSecuriTEE and TRNG	V1.0 Date: 2025-03-21

Table 3 Included Guidance Documents

1.6. Platform Functional Overview and Description

1.6.1. Platform Type

The RealSecuriTEE is a Root of Trust (RoT) silicon IP core specifically designed to secure System-on-Chip (SoC) platforms and their operations. By providing a trusted foundation, it enables secure boot, protects sensitive key material and assets, and ensures the integrity of IoT, gateway, and edge devices.

The RealSecuriTEE features a dedicated set of hardware resources, including CPUs and memory, which create an isolated and secure enclave. This secure enclave supports a range of security functions, including hardware-based cryptographic acceleration, random number generation, key management, and secure debugging. Additionally, the RealSecuriTEE allows for the optional encryption of flash memory contents, which can be decrypted on-the-fly during runtime, thereby providing an additional layer of protection for sensitive data and algorithms.

1.6.2. Physical Scope

The platform scope is indicated by the red border in Figure 1. The other parts of Figure 1, comprising the FPGA processing system, are outside of the evaluation scope. Please note that the green area may be adjusted according to the semiconductor process used in future products, and the OTP Macro implementation may vary depending on the IP vendor.

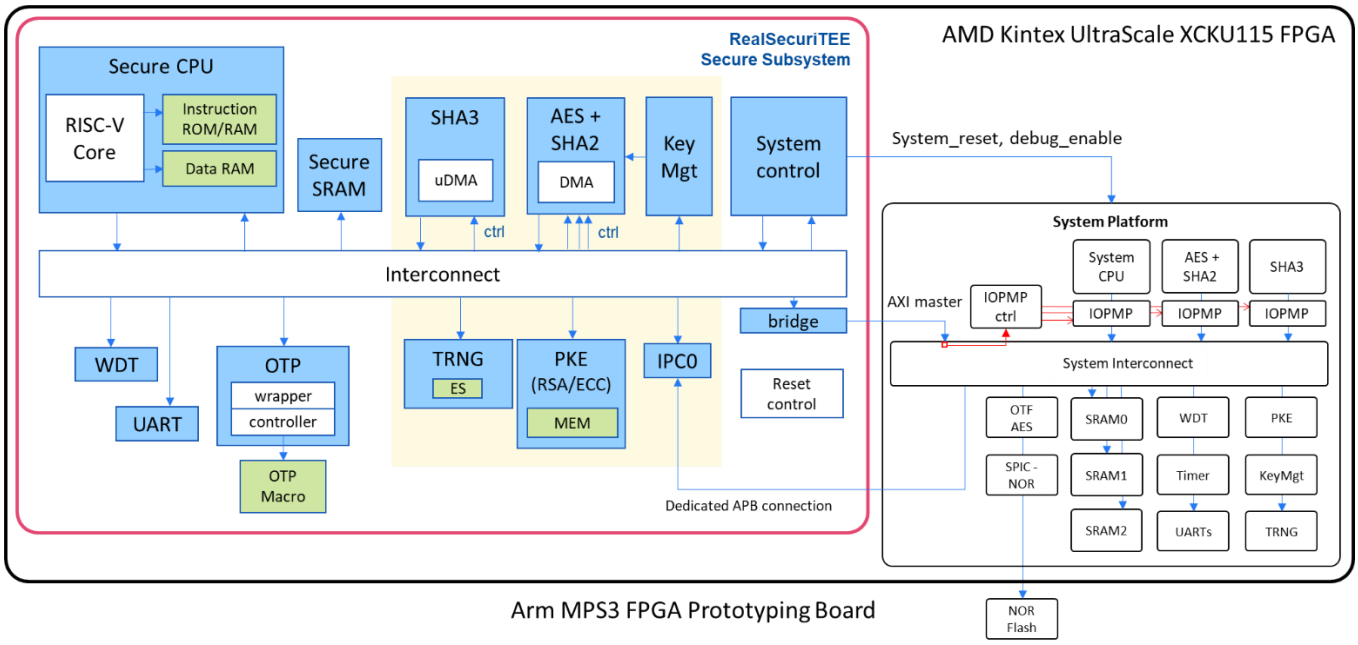

Figure 1 RealSecuriTEE TOE scope

1.6.3. Logical Scope

The logical scope listed as below. Any additional firmware, OS or application software stored on the platform is not in scope of this evaluation.

Type	Name	Release	Description
IP Hardware	RealSecuriTEE	11	Silicon IP. RTL HW IP.

ROM Firmware	RealSecuriTEE ROM	100	ROM Firmware
Secure boot code	BL2 FW	1.0.0	The secondary boot code validates firmware and certificate and run in Internal Memory (IMEM) and Data Memory (DMEM).
Secure Firmware	Siren FW	1.0.0	The secure firmware that provides various secure services, running in Secure SRAM.

1.6.4. Life Cycle

The life cycle (LC) is managed by the platform, see [FW_Manual] section 5.1 Table15 for further information. The LC states are listed below.

LC State	Description
Development	This initial state serves as a baseline for the development and testing of RealSecuriTEE, providing a foundational environment for the creation and validation of the system.
Manufacture	At this stage, trust provisioning can be used to burn in the assets required for the product, such as keys and firmware.
Deployment	In this state, RealSecuriTEE is in a secure mode, where all test and debug interfaces are disabled, preventing unauthorized access. Meanwhile, non-secure software development can proceed without interruption. However, to access RealSecuriTEE, a debug authentication process must be performed.
Decommission	When a chip needs to be scrapped and is no longer in use, developers should set the lifecycle state to decommission and erase all data on the chip. At this point, all debug ports are permanently disabled and cannot be used.

As shown in Figure 2, the Life cycle Diagram, the platform's initial lifecycle state is the Development state. Upon completion of all development and testing, the lifecycle state is transitioned to the Manufacture state using the OTP provision. Once all assets have been properly configured, the lifecycle state is further transitioned to the Deployment state, again using the OTP provision. Finally, when the chip is no longer in use, the lifecycle state can be switched to Decommission using an unlock certificate. It is important to note that the lifecycle state is non-reversible, meaning it cannot be reverted to a previous state.

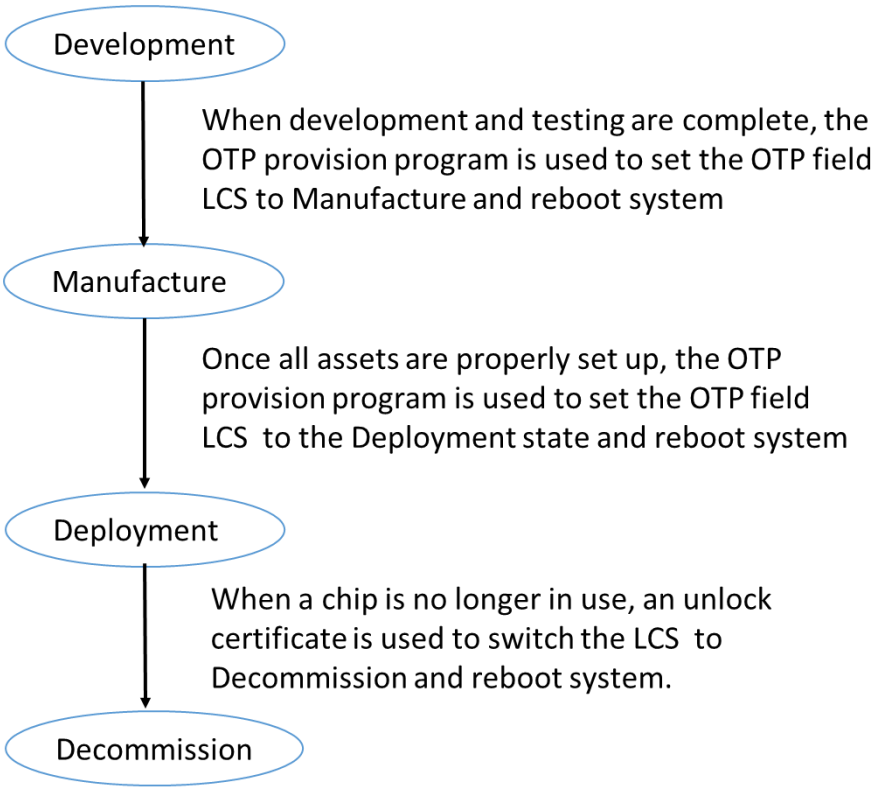

Figure 2. Lifecycle diagram

1.6.5. Usage and Major Security Features

The main security feature of the TOE are as follows:

- Secure boot

  Ensures the authenticity, integrity, and confidentiality of the BL2 FW, Siren FW, and other software or certificates during the boot process. To verify the integrity of the image, the image is first signed using the RSA-2048 (PSS) algorithm. During the boot process, the image's signature is verified using the RSA-2048 (PSS) algorithm. However, prior to signature verification, the image itself is encrypted and must be decrypted to plaintext. The image is decrypted using AES-128 in counter mode, but the AES key used for decryption is itself encrypted using the RSA-2048 (OAEP) algorithm. Therefore, the AES key must first be decrypted using the RSA-2048 (OAEP) private key to obtain the plaintext key. Subsequently, the plaintext key is used to decrypt the image to plaintext, and finally, the image's signature is verified. For additional information, please refer to Section 3.2.4
- Cryptographic functions

  The platform provides the cryptographic service for securely doing cryptographic operations. In addition, since keys are assets, a key management (see FW_Manual section 8.2.1) mechanism is used

to protect keys requiring safeguarding and safely stored in TOE or flash. Applications don't need to access the key material directly, but instead use opaque key identifiers.

- Firmware update and recovery
  - Secure Update is a secure process for updating firmware image to a new version, ensuring confidentiality, integrity, and authenticity. The new firmware image is pre-signed with RSA-2048 (PSS) and encrypted with AES-128 in Counter (CTR) mode. The AES encryption key is itself encrypted with RSA-2048 (OAEP) using a separate RSA public key. During Secure Boot, the encrypted AES key is decrypted using the RSA private key pre-programmed into One-Time Programmable (OTP) memory, and then the decrypted AES key is used to decrypt the firmware image. The decrypted image is then verified using RSA-2048 (PSS), which ensures its integrity and authenticity by confirming that the image has not been tampered with. For additional information, please refer to Section 3.2.6.
  - Firmware recovery is a backup mechanism that stores two firmware images in the flash memory, utilizing two separate slots: Slot A for the active image and Slot B for the backup image. During the secure boot process, the system attempts to boot using the active image in Slot A. If the boot process fails, the system automatically switches to the backup image in Slot B to attempt a successful boot. If both images fail to boot, the system will be unable to complete the boot process. For additional information, please refer to [FW_Manual] section 7.2.
- Anti-rollback is a security mechanism designed to ensure that the TOE can only be upgraded to a higher version of firmware or software, and not downgraded to a previous version, thereby preventing known vulnerabilities or security weaknesses from being reintroduced. The TOE utilizes several non-volatile (NV) counters to track firmware image versions, which are stored in One-Time Programmable (OTP) memory to prevent older versions from being programmed into the TOE's flash memory. For additional information, please refer to [FW_Manual] section 3.5. The anti-rollback check process is as follows:
  - Validation Criteria: The anti-rollback check passes if the version of the firmware image to be updated or verified is greater than or equal to the anti-rollback version counter stored in the OTP field. Conversely, if the version is lower, the check fails, and the TOE enters an infinite sleep mode or panic state.
  - Updating the OTP Field: Upon successful anti-rollback validation during the secure boot process, the anti-rollback version counter in the OTP field is updated to reflect the current firmware image version. This updated value serves as the latest acceptable image version, ensuring that any subsequent firmware updates must be equal to or higher than this version.
- Secure storage
  - All keys or assets stored in the TOE via the cryptographic service are defined with the ownership and use policies.

- All user keys or assets stored outside of TOE, such as flash, will be encrypted via the cryptographic service and have been defined with the ownership and use policies - cryptographic keystore.
- All user data stored outside of TOE, such as flash, will be encrypted through the encryption mechanism in the platform – secure encrypted storage.

- Device Attestation

  The Attestation provides a proof to a remote party on the platform's genuine identity, its software and firmware versions, as well as its integrity and life cycle state. For additional information, please refer to Section 3.2.3.
- Lifecycle control

  Lifecycle management refers to the security states and transition conditions throughout the entire lifecycle of the TOE, from development and testing to decommissioning. For a detailed description of these states, please refer to Section 1.6.4.
- Secure Debug

  Secure Debug protects access to debugging features through unlock certificate authentication using the RSA-2048 (PSS) algorithm, requiring a new certificate for each use since they are single-use. For further details, please refer to Section 3.3.1.

1.6.6. Required Hardware/Software/Firmware

In addition to RealSecuriTEE, the following system platforms are also required:

Name	Description
System platform	Non-secure system platform includes system CPU, SRAM, SPIC, etc.
External NOR FLASH	Provides a storage solution for all software components, including firmware images and user-specific data
NS Firmware	The non-secure firmware that runs on the system platform.

For more details please refer to [FW_Manual] section 1.3.2 and section 1.3.3

2. Security Objectives for the Operational Environment

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

Title	Description	Reference

Secure Debug	The integrating environment is expected to configure the debug functionality as described in [FW_Manual] section 5.2 table 17 and table 18.	[FW_Manual] section5.2 table 17 and table 18.
Ensure UUID Uniqueness	A UUID, also known as an Implementation ID, is embedded in the platform to identify the chip manufacturer and/or the finished device. For RealSecuriTEE, the Implementation ID is identical to the RTL HW IP's UUID.	This document [FW_Manual] section 10.1 table 26
Secure key Provision	Keys shall be provisioned for the corresponding security functions, including: attestation, image authentication, encryption, and secure debug.	[FW_Manual] Section 6.1 table19
Trusted Users	Actors in charge of platform management, for instance for signature of firmware update, are trusted.	This document
Lifecycle Management	The RealSecuriTEE provides lifecycle states and a secure mechanism for lifecycle state transition according to the use case, and the operational environment is expected to configure the platform accordingly for lifecycle state transitions. In general, the RealSecuriTEE is expected to be configured in Deployment state before leaving production.	[FW_Manual] section 5.1
Cryptographic Algorithm and Key Length	The operating system or application code are expected to select an appropriate algorithm and key length set to fulfill the security requirement for the intended use case.	This document
DRBG Use	The entropy source (e.g. TRNG) shall be used as the seed source for DRBG	[FW_Manual] section 8.2.3

3. Security Requirements and Implementation

3.1. Security Assurance Requirements

The SESIP claimed assurance requirements package is SESIP2 as described in Section 4.1.

3.1.1. Flaw Reporting Procedures (ALC_FLR.2)

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

Realtek has CTC Security Team (CTCST) which is responding to, manage, investigate, receive and release information about security issues regarding Realtek products.

CTC Security Team (CTCST) follows the guidelines outlined in the following documents:

- Realtek Vulnerability Disclosure Guidelines [RUDG_GUID] – for the disclosure of security vulnerabilities
- Realtek Responsible Disclosure Policy [RRDP_GUID] – for vulnerability reporting procedures
- Realtek Product Vulnerability Handling Process [RPVHP_GUID] – for the process of handling security vulnerabilities

For external parties that that wish to report a vulnerability, they may contact Realtek via the link below:
https://www.realtek.com/Article/Index?menu_id=848

3.2. Base PP Security Functional Requirements

In the following Security Functional Requirements, the term platform covers the RealSecuriTEE physical and logical scope, and the term App refer to any additional firmware, OS or application software which is out of evaluation scope. It represents a part of the final connected device. RealSecuriTEE fulfils the following security functional requirements:

3.2.1. Verification of Platform Identity

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

Conformance rationale:

The identification for different parts of the platform can be done by using siren_read_impl_id OTP provision command as specified in [FW_Manual] sections 6.3 or psa_initial_attest_get_token secure service command as specified in [FW_Manual] sections 10.3. The returned values shall be the same as the values indicated Section 1.6.3 more specifically,

- IP Hardware: Bits 32-39 of the Implementation ID can be retrieved using either the siren_read_impl_id command or the psa_initial_attest_get_token command.
- ROM Firmware: Bits 0-31 of the Implementation ID can be retrieved using either the siren_read_impl_id command or the psa_initial_attest_get_token command.
- Secure boot code (BL2 FW): The psa_initial_attest_get_token command can be used with the software

component type is "BL" and version "v1.0.0" to identify the Secure Bootloader.

- Secure Firmware (Siren FW): The psa_initial_attest_get_token command can be used with the software component type is "PROT" and version "v1.0.0" to identify the Secure Firmware.

NOTE that the implementation ID is a unique 256-bit device identifier, which is stored as the IP UUID. This UUID contains detailed platform information, including version details, as specified in [FW_Manual] section 6.1. Note that the RealSecuriTEE is not a production-ready product, and the UUID definition may differ across various products.

3.2.2. Verification of Platform Instance Identity

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

Conformance rationale:

During the attestation process, the platform derives the initial attestation public key from the initial attestation private key stored in the One-Time Programmable (OTP) memory and generates a 32-byte hash value using SHA2-256, which serves as the instance ID. However, to ensure the security and integrity of the attestation process, the verification of the instance ID requires a private key signature. Specifically, the platform uses the private key to sign an attestation token, which can then be verified by the user using the corresponding public key. This ensures that only the legitimate platform can generate a valid attestation token, as the private key is only accessible to the platform. By requiring a private key signature, the attestation process provides a robust and secure mechanism for verifying the authenticity and integrity of the instance ID. This is clear and well described in section 3.2.3. For additional information, please consult [FW_Manual] section 10.1.

The complete instance ID information can be retrieved using the psa_initial_attest_get_token secure service command as specified in [FW_Manual] sections 10.3.

3.2.3. 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.

Conformance rationale:

During the manufacturing phase, as described in section 1.6.4, Realtek's secure device enables the creation of initial attestation keys, providing cryptographic proof of the device's origin and secure provisioning of their own assets. A unique initial attestation private-public key pair is generated for the platform, with the

private key securely stored in the platform's internal OTP memory. The corresponding public key is signed by Realtek's secure device and provided as a certificate, which serves the actual proof of the platform's origin.

To ensure the integrity of the platform, it is essential to transition the life cycle state from manufacturing to deployment. This guarantees that any attempt to clone or modify the platform will be detectable. The Siren FW (secure firmware) supports the PSA attestation API to produce an initial attestation token in IETF EAT format containing measurements of the firmware, which provides an attestation service which reports on the device identity, firmware measurements and run-time state of the device. The attestation can be verified by remote entities. The tokens are signed using the initial attestation private key stored in the internal OTP. For more detailed information on the attestation process and its implementation, please refer to [FW_Manual] section 10.3, Figure 49.

3.2.4. 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.

Conformance rationale:

Secure boot prevents unauthorized code from executing on the designated product. It achieves this security level by having the device ROM verify the first executable image that runs after reset. Specifically, the ROM loads the initial executable image from external flash memory into the platform's internal memory, decrypting the encrypted image, encrypted to ensure confidentiality, before verifying the authenticity of the code. Upon successful verification, control is transferred to the verified image, establishing a chain of trusted code from the ROM to the boot code. This chain of trust can be extended further to additional software layers through digital signature verification.

The platform's ROM boot code always executes immediately following a reset to provide secure boot functionality. The firmware image intended for loading is signed using either the RSA-2048 (PSS) algorithm, with the signature appended at the end of the firmware image. Additionally, the firmware within the image is encrypted using AES-128 in Counter (CTR) mode. For details regarding the firmware image format, please refer to [FW_Manual] Section 2.1.

The complete firmware image contains additional information beyond the firmware itself, such as headers. Most importantly, it includes a public key used for verifying the RSA signature and an encrypted AES key,

which is necessary for decrypting the encrypted firmware. For further details, please see [FW_Manual] Section 3.1.

If the ROM boot code fails to verify the firmware image, the platform may enter an infinite sleep mode. When the platform enters an infinite sleep mode, all services, including the secure update service, become inaccessible. The system halts execution, and no further operations are possible. Nevertheless, the developer can still recover the platform by updating the image stored in the flash memory. For additional information, refer to [FW_Manual] Section 3.1.

3.2.5. 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.

Conformance rationale:

The attestation token provides the security lifecycle state of the platform, please see section 1.6.4.
For more detailed information, please refer to the [FW_Manual], section 10.1, Table 26.

3.2.6. 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.

Conformance rationale:

Secure Update is a secure process for updating firmware to a new version. The firmware includes BL2 FW and Siren FW (refer to Section 1.6.3), as well as Non-Secure Firmware (NS FW) running on the non-secure system. A key certificate is used to verify the NS FW, and thus the Secure Update process provides a mechanism for updating the platform securely.

Firmware images or certificates are encrypted using AES-128 and signed using the RSA-2048 (PSS) or RSA-3072 (PSS) algorithm, conforming to the platform firmware image format described in [FW_Manual] Section 2.1. Secure Update ensures the authenticity, integrity and confidentiality of the image. Additionally, it ensures the image is the latest version, preventing rollback to older firmware. For more detailed information, please refer to the [FW_Manual] section 7.1.

The platform's Siren FW provides the Secure Update service. When the platform receives a Secure Update command (refer to [FW_Manual] Section 7.1.1 for details), the Secure Update service writes the new firmware image into flash memory. Following this, the platform performs a reboot and initiates the Secure

Boot process, which handles the decryption and signature verification of the updated firmware image. In summary, the Secure Update service is responsible for writing the firmware image into flash memory, while Secure Boot handles image decryption and signature verification.

However, in certain cases, the secure update cannot be used, including the following scenarios:

- The lifecycle state is Decommission.
- The lifecycle state is in Development, Manufacture, or Deployment, and the platform encounters a secure boot failure due to image validation failure, causing the platform to enter an infinite sleep mode.

Except for these situations where secure update cannot be used, the platform's Secure Update process must fulfill the following security requirements:

- Integrity and Authenticity: The new firmware image must be pre-signed using RSA-2048 (PSS) or RSA-3072 (PSS) algorithms, allowing verification of its integrity and authenticity during the Secure Boot process. The hash of the trusted root key used for RSA signature verification is securely pre-programmed into OTP.
- Confidentiality: The new firmware image is encrypted using AES-128 in Counter (CTR) mode. The AES encryption key itself is encrypted with RSA-2048 (OAEP) using a separate RSA public key or ECIES-P256 with a separate ECIES (secp256r1) public key. The encrypted AES key is stored in flash memory along with the encrypted firmware image. During Secure Boot, the RSA private key or ECIES private key (SFB_KDK) pre-programmed into OTP is used to decrypt the AES key, which then decrypts the firmware image. For more details for RSA private key or ECIES private key (SFB_KDK), please refer to Section 3.2.11.
- Anti-Rollback: The platform provides several non-volatile (NV) counters, such as NV_CNT_BL for tracking BL2 firmware versions, stored in OTP to prevent older firmware versions from being programmed into the platform's flash memory. For more details, please refer to [FW_Manual] Section 3.5

3.2.7. Software Attacker Resistance: Isolation of Platform

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.

Conformance rationale:

RealSecuriTEE is the secure platform (SPE). The other part of the system, including Non-secure (NS) CPU and Non-secure Memory (NS-SRAM 0/1/2) is in the NSPE. For more details, please refer to Figure 1. The platform operates on dedicated hardware resources which is isolated from the CPU which runs the application by SoC design. The sensitive information like secret key, or user data are processed in platform isolated hardware resources. For the secret keys pre-defined or run-time generated, they are stored on platform OTP, or platform memory correspondingly. Application run on system CPU cannot access platform OTP, platform memory, and platform cryptographic engine by SoC design. Platform uses its own cryptographic engines and OTP macro located on its sub-system bus fabric.

NSPE exchanges data with the platform via Non-secure SRAM 0/1/2, and the NSPE requests secure function via Inter-Processor Communication call (IPC and psa_call interfaces). See more details in [FW_Manual] section 1.3.2 and section 1.3.3 for non-secure SRAM 0/1/2 and see more details in [FW_Manual] section 1.2.1 for Inter-Processor Communication. The platform implements memory access control to prevent the NSPE from extracting sensitive information from the platform memory through secure function calls. See more details of memory access control in [FW_Manual] section 4.4.4. The platform image and user data stored on external flash are encrypted to protect sensitive data according to the implementation of Firmware Update and Secure Encrypted Storage. See more details in [FW_Manual] section, 9.2 for user data encryption. See more details in section 3.2.6 for image encryption.

3.2.8. Cryptographic Operation

The platform provides Operations in functionality with algorithms in Table 4 as specified in specifications in Table 4 for key lengths described in Table 4 and modes described in Table 4.

Conformance rationale:

Algorithm	Key Lengths (bits)	Modes	Specification	Usage
AES	128, 192, 256	ECB, CBC, CTR, CFB, OFB	NIST FIPS 197 SP 800-38A	Encryption, decryption
	128, 192, 256	CMAC	SP 800-38B	MAC
	128, 192, 256	CCM	SP 800-38C	Authenticated encryption with associated data (AEAD) and authenticated decryption

	128, 192, 256	GCM, GMAC	SP 800-38D	Authenticated encryption with associated data (AEAD) and authenticated decryption
SHA2	-	Digest length: 224, 256, 384, 512	NIST FIPS 180-4	Message digest
SHA3	-	Digest length: 224, 256, 384, 512	NIST FIPS 202	Message digest
RSA	2048, 3072, 4096	PSS	NIST FIPS 186-4 NIST FIPS 186-5	Signature generation and verification
		PKCS#1v1.5		
	2048	OAEP	SP 800-56B	Encryption, Decryption
ECDSA	P-224, P-256, P-384	-	NIST FIPS 186-4 NIST FIPS 186-5	Signature generation and verification
HMAC	8 to 512K	SHA2	NIST FIPS 198	Key derivation
ECDH	P-256, P-384	-	SP 800-56A	Shared secret computation

Table 4. Cryptographic Operation

The cryptographic operations supported by Lalu hardware engine, obtained CAVP certification: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/details?product=18474

3.2.9. Cryptographic Random Number Generation

The platform provides a way based on DRBG to generate random numbers to as specified SP800-90A.

Conformance rationale:

The Deterministic Random Bit Generator (DRBG) is based on NIST SP800-90A standard and the seed needs to be from an entropy source. The user will use DRBG to generate random number, which is implemented based on NIST SP 800-90A.

3.2.10. Cryptographic Key Generation

The platform provides a way to generate cryptographic keys for use in ECDSA, AES and HMAC as specified in Table 5, and for key lengths described in Table 5.

Conformance rationale:

The platform supports ECDSA key generation through siren firmware [FW_Manual] section 1.2.2. By execution of this command, ECDSA key pair can be generated.

Users can use the key derivation function to derive keys according to the MAC based Key Derivation Function [FW_Manual] section 8.2.2.

Users can use provided random function to generate AES symmetric keys [FW_Manual] section 8.2.3.

Algorithm	Specification	Key Lengths
ECDSA	NIST FIPS 186-4 NIST FIPS 186-5	P-224, P-256, P-384
Symmetric	None	128, 192, 256

Table 5. Cryptographic Key Generation

3.2.11. Cryptographic Keystore

The platform provides a way to store cryptographic keys listed in Table 6 such that not even the application can compromise the integrity, authenticity, confidentiality of this data. This data can be used for the cryptographic operations listed in Table 6.

Key names	Location	Description	Cryptographic Operation
HUK	OTP	The root key derivation key of the platform. Immutable, cannot be exported outside the platform.	Key derivation
IAK	OTP	Initial attestation key of platform. Immutable, cannot be exported outside the platform.	ECDSA
FW_DK	Flash	Firmware Decryption Key. The symmetric key to decrypt firmware images during secure boot, which is stored encrypted in unsecure flash.	AES-CTR decryption
SFB_KEK	OTP	Secure Firmware Boot Key Encryption Key, which is used for decrypting the FW_DK for	RSA/ECIES decryption

		secure FW encryption. Cannot be exported outside the TOE.	
User defined keys	Secure SRAM or Flash	User defined keys imported from an application. User defines whether the key can be exported or not.	Key derivation Encryption, decryption, authentication

Table 6. Cryptographic KeyStore

Conformance rationale:

Keys (assets) are stored in OTP, secure SRAM or Flash, depending on the usage of the keys. User defined keys may be imported or generated by applications, and used for the cryptographic operations encryption, decryption, signature generation, MAC generation and verification, key derivation, shared secret generation.

In OTP, keys are securely stored in advance, are immutable and cannot be exported or accessed by NSPE. The keys stored in OTP are only for internal use, and there is no way for applications to store keys in the OTP.

Moreover, the platform provides a dedicated flash area for cryptographic key storage. If an application intends to store a key within the flash memory and delegate its management to the TOE, it is mandatory to import or generate the key via the cryptographic service inside of TOE. The intended use of each key, including its usage policy, exportability and ownership, must be tracked through the management of key attributes (see more in [FW_Manual] Section 8.2.1). For example, the PSA_KEY_USAGE_EXPORT flag indicates that the key can be exported to application via the cryptographic service. The PSA_ALG_GCM flag indicated that the key is used for AES-GCM algorithm. The PSA_KEY_TYPE_DERIVE flag indicates that the key is used for key derivation, and cannot be used for other cryptographic operations. Keys or assets stored in this dedicated flash area are encrypted with AES-GCM AEAD with a run-time derived key (see [FW_Manual] Section 9.1.5) via the cryptographic service, random IV and security attributes for owner and key usage policy. This run-time derived key, will be reproduced using the same parameters upon the application's subsequent request, which does not persist permanently within the memory.

It is noteworthy that the purpose and storage mechanism in which FW_DK resides in flash differ from the previously outlined descriptions. FW_DK is stored in each firmware image in flash, and is encrypted with RSA-OAEP with SFB_KEK. The firmware image will be programmed by an application. Once FW_DK

is successfully decrypted, FW_DK will be used to decrypt the firmware image with AES-CTR during secure boot (see the boot flow in [FW_MANUAL] section 3).

At last, keys stored in secure SRAM is a volatile memory inside of TOE. Following the same key management approach as described earlier, the attributes of each key are recorded for management purposes via the cryptographic service. A notable distinction is that keys residing in secure SRAM are not encrypted since secure SRAM is the TOE memory, applications cannot access directly. Furthermore, due to the volatile nature of SRAM, these keys are automatically erased upon power loss. However, applications can also manually destroy them by using the psa_destroy_key command.

3.3. Optional Security Functional Requirement

3.3.1. Secure Debugging

The platform provides JTAG interface authenticated as specified in [FW_Manual] Section 5.2 with debug functionality. The platform ensures that all data stored by the application, with the exception of non-sensitive data, is made unavailable.

Conformance rationale:

The availability of debug functionality depends on the life-cycle state of the platform. This is enforced by life-cycle management of the platform as specified in [FW_Manual] Section 5.1. For example, in Development and Manufacture states, access to debug functionality is always possible. On the other hand, the same access is completely disabled in Decommission and Unknown state. In the other life cycle states, the deployment states, access to debug functionality is enabled, but shall be authorized.

The platform enforces a debug authentication protocol, which requires a valid unlock certificate issued by Realtek. This unlock certificate must be verified by the platform in order to gain debug access. The verification mechanism is based on an RSA-2048 (PSS) algorithm, and the unlock certificate is single-use, requiring a new certificate to be issued for each use. For more information, please refer to [FW Manual] Section 5.2.

3.3.2. Secure Encrypted Storage

The platform ensures that all data stored by the application is encrypted by AES-GCM by the HMAC-based derived per-device unique key of key length 128 bits.

Conformance rationale:

The per-device unique AES key is runtime-generated in platform internal memory during the execution of the psa_ps_set or psa_ps_get command. And a proprietary key label of 96 bit is added to derivation. File ID is used as the additional data of AES-GCM algorithm. The Initial Vector (IV) is 96 bits. See more in [FW_Manual] section 9.2.1.

The application needs to specify the file ID, the data buffer address, and the data length of the data to be stored through psa_ps_set. The platform then acquires the user data into the platform memory, encrypts, then stores cipher text to the external flash. When application needs to get the data, the platform reads back the data, decrypt and return it to the application through psa_ps_get API. See more in [FW_Manual] section 9.2.2

3.4. Compliance Functionality

3.4.1. Reliable Index

The platform implements a strictly increasing function.

Conformance rationale:

The platform provides Monotonic Counters Services. The platform has 256-bit monotonic counters that can be read and only incremented until saturation. See more in [FW_Manual] section 3.5.

4. Mapping and Sufficiency Rationales

This ST and associated TOE provide exact conformance to SESIP Profile for PSA Certified Level 2, aiming at both SESIP certificate and PSA certificate.

4.1. Assurance

Assurance Class	Assurance Family	Covered By
ASE: Security target evaluation	ASE_INT.1 ST Introduction	Section 1
	Rationale: The SESIP profile reference is in Section 1.2	

	The TOE reference is in Section 1.3 The TOE overview and description in Section 1.6.	
	ASE_OBJ.1 Security requirements for the operational environment	Section 2
	Rationale: The objectives for the operational environment in Section 2 refer to the guidance documents.	
	ASE_REQ.3 Listed security requirements	Section 3
	Rationale: All SFRs in this ST are taken from [1]. SFR "Identification of Platform Type" is included. SFR "Secure Update of Platform" is mentioned but refers to ALC_FLR.2.	
	ASE_TSS.1 TOE Summary Specification	Section 3
	Rationale: All SFRs are listed per definition, and for each SFR the implementation and verification are defined in the SFR.	
ADV: Development	ADV_FSP.4 Complete functional specifications	Section 1.5
	Rationale: The evaluator will determine whether the provided evidence is suitable to meet the requirement.	
AGD: Guidance documents	AGD_OPE.1 Operational user guidance	Section 1.5
	Rationale: The evaluator will determine whether the provided evidence is suitable to meet the requirement.	
	AGD_PRE.1 Preparative procedures	Section 1.5
	Rationale: The evaluator will determine whether the provided evidence is suitable to meet the requirement.	
ALC: Life-cycle support	ALC_FLR.2 Flaw reporting procedures	Section 3.1.1
	Rationale: The flaw reporting and remediation procedure is described.	
ATE: Tests	ATE_IND.1 Independent testing: conformance	Material provided to evaluator.
	Rationale:	

	The evaluator will determine whether the provided evidence is suitable to meet the requirement.	
AVA: Vulnerability Assessment	AVA_VAN.2 Vulnerability analysis	Vulnerability assessment is performed by the evaluator
	Rationale: The evaluator will determine whether the provided evidence is suitable to meet the requirement.	

Table 7. Assurance Mapping and Sufficiency Rationales