Advanced Micro Devices (AMD)

AMD ASP Cryptographic CoProcessor ("Strix Halo")

FIPS 140-3 Non-Proprietary Security Policy

Prepared by:

atsec information security corporation

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

www.atsec.com

Document version: 1.1

Last update: 2025-10-27

Table of Contents

List of Tables

List of Figures

1 General

1.1 Overview

This document is the non-proprietary FIPS 140-3 Security Policy for the AMD ASP Cryptographic CoProcessor ("Strix Halo") cryptographic module. It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-3 (Federal Information Processing Standards) for an overall Security Level 1 module.

This Non-Proprietary Security Policy may be reproduced and distributed, but only whole and intact and including this notice.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	1
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The AMD ASP Cryptographic CoProcessor ("Strix Halo”) cryptographic module (hereafter referred to as “the module”) is defined as a sub-chip hardware module in a single chip embodiment, with hardware and firmware components implementing general purpose cryptographic algorithms.

Module Type: Hardware

Module Embodiment: Single Chip

Module Characteristics: SubChip

Cryptographic Boundary:

The module consists primarily of the ARM Cortex-A5, Random Bit Generation hardware, Security Infrastructure Block, Cryptographic CoProcessor, and OTP fuses. These hardware components are sub-components of the “IOD” (Ryzen RIOD2.0), which itself is a smaller die in the larger single chip embodiment, the Ryzen SoC.

OTP fuses are used to persistently store FIPS support enablement and versioning information, security state information, end-of-life information, and Entropy Source configuration values (sample rate, sample count, RCT and APT cutoffs).

In addition, there is a ROM firmware component (“libROM”) permanently stored inside the Ryzen SoC, and an overlay firmware component (“overlay firmware”) permanently stored inside SPI flash storage, outside the SoC, which is loaded into the SRAM on startup.

The block diagram in Figure 2 shows the design of the module when the module is operational and the firmware components are loaded into the SRAM. In this diagram, the physical boundary of the module, defined by the perimeter of the Ryzen SoC (i.e., the enclosure of the SoC), is indicated by a dashed purple line. The cryptographic boundary is represented by the components painted in orange blocks. Solid orange lines indicate the flow of data within the cryptographic module (i.e., internal paths). Dashed green lines are used to denote the logical interfaces defined in Section 3.

Components in white are only included in the diagram for informational purposes. They are not included in the cryptographic boundary (and therefore not part of the module’s validation).

Tested Operational Environment’s Physical Perimeter (TOEPP):

The TOEPP is the Ryzen SoC (shown in Figure 1), a rectangular enclosure measuring approximately 37.5 mm x 45.0 mm x 1.59 mm.

Figure 1: Ryzen AI Max+ PRO 395

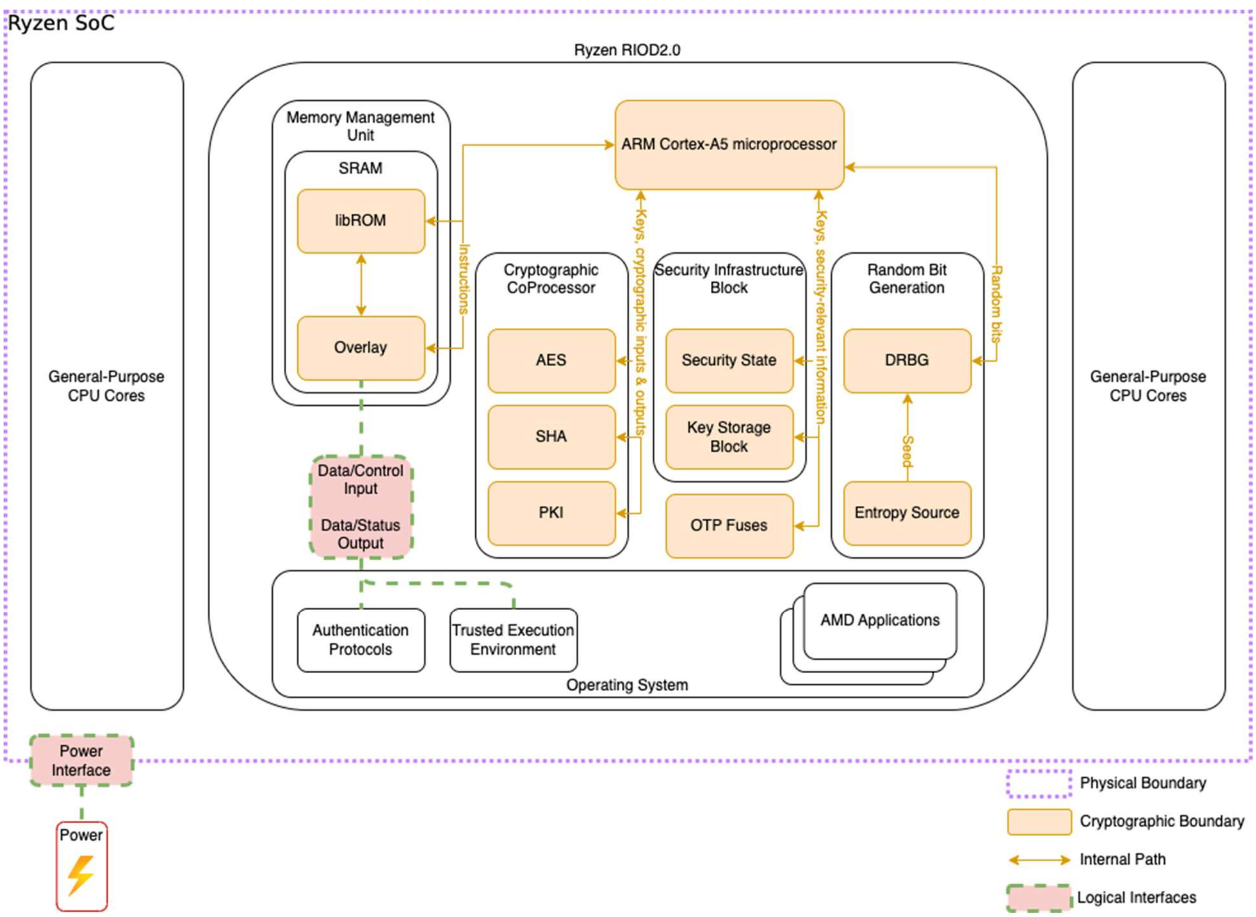

Figure 2: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

Model and/or Part Number	Hardware Version	Firmware Version	Processors	Features
Ryzen AI Max+ PRO 395	A0-1-4004	00400008	ARM Cortex-A5	N/A

Table 2: Tested Module Identification – Hardware

2.3 Excluded Components

There are no components excluded from the requirements of the FIPS 140-3 standard.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved mode	Automatically entered whenever an approved service is requested	Approved	Equivalent to the indicator of the requested service (FipsIndicatorStatus is set to 2)
Non-approved mode	Automatically entered whenever an non-approved service is requested	Non-Approved	Equivalent to the indicator of the requested service (FipsIndicatorStatus is not set to 2)

Table 3: Modes List and Description

After passing all pre-operational self-tests and conditional self-tests executed on startup, the module automatically transitions to the approved mode. No operator intervention is required to reach this point. In the operational state, the module accepts service requests from calling applications through its logical interfaces. The operator can verify that the module is operational by requesting the RL_ARCL_GetState service and comparing the returned ArclState value with 4.

Mode Change Instructions and Status:

The module automatically switches between the approved and non-approved modes depending on the services requested by the operator. The status indicator of the mode of operation is equivalent to the indicator of the service that was requested.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6730	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A6730	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A6730	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A6730	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A7030	Direction - Encrypt Key Length - 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
Conditioning Component AES-CBC-MAC SP800-90B	A5337	Key Length - 256	SP 800-90B
Counter DRBG	A7030	Prediction Resistance - No Mode - AES-256 Derivation Function Enabled - No	SP 800-90A Rev. 1
HMAC-SHA-1	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A6730	Key Length - Key Length: 112-524288 Increment 8	FIPS 198-1
KDF SP800-108	A6730	KDF Mode - Counter Supported Lengths - Supported Lengths: 112-4096 Increment 8	SP 800-108 Rev. 1
RSA SigVer (FIPS186-2)	A6730	Signature Type - PKCSPSS Modulo - 1536	FIPS 186-4
RSA SigVer (FIPS186-4)	A6730	Signature Type - PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-5)	A6730	Modulo - 2048, 3072, 4096 Signature Type - pss	FIPS 186-5
SHA-1	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Algorithm	CAVP Cert	Properties	Reference
SHA3-224	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A6730	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHAKE-128	A6730	Output Length - Output Length: 1344	FIPS 202
SHAKE-256	A6730	Output Length - Output Length: 1088	FIPS 202

Table 4: Approved Algorithms

Vendor-Affirmed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

Name	Caveat	Use and Function
AES-ECB using the non-approved key	Allowed per IG 2.4.A example 2	De-obfuscate data using the non-approved RTL key for a non-security relevant purpose.

Table 5: Non-Approved, Allowed Algorithms with No Security Claimed

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
HMAC with key lengths less than 112 bits	Message authentication
RSA (pre-hashed message)	Signature verification
SHA-384 with non-standard initial hash value	PCR-based memory measurement
CCP_HAL algorithm	Message digest (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512), XOF (SHAKE128, SHAKE256), encryption, decryption (AES, ECB, CBC, OFB, CFB, CTR, GCTR, IAPM, XTS), message authentication (AES CMAC)
SIB_HAL algorithm	Random number generation using the DRBG.

Table 6: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Encryption	BC-UnAuth	Encrypt a plaintext		AES-CBC: (A6730) AES-CTR: (A6730) AES-ECB: (A6730)
Decryption	BC-UnAuth	Decrypt a ciphertext		AES-CBC: (A6730) AES-CTR: (A6730) AES-ECB: (A6730)
Message digest	SHA	Compute a message digest		SHA-1: (A6730) SHA2-224: (A6730) SHA2-256: (A6730) SHA2-384: (A6730) SHA2-512: (A6730) SHA3-224: (A6730) SHA3-256: (A6730) SHA3-384: (A6730) SHA3-512: (A6730)
XOF	XOF	Compute an extendable output message digest		SHAKE-128: (A6730) SHAKE-256: (A6730)
MAC	MAC	Compute a MAC tag		AES-CMAC: (A6730) HMAC-SHA-1: (A6730) HMAC-SHA2-224: (A6730) HMAC-SHA2-256: (A6730) HMAC-SHA2-384: (A6730) HMAC-SHA2-512: (A6730) HMAC-SHA3-224: (A6730) HMAC-SHA3-256: (A6730)

Name	Type	Description	Properties	Algorithms
				HMAC-SHA3-384: (A6730) HMAC-SHA3-512: (A6730)
Random number generation	DRBG	Generate random bytes		Conditioning Component AES-CBC-MAC SP800-90B: (A5337) AES-ECB: (A7030) Counter DRBG: (A7030)
Key derivation	KBKDF	Derive a key from a key derivation key		KDF SP800-108: (A6730)
Signature verification	DigSig-SigVer	Verify a digital signature		RSA SigVer (FIPS186-5): (A6730)
Signature verification (legacy)	DigSig-SigVer	Verify a digital signature		RSA SigVer (FIPS186-2): (A6730) RSA SigVer (FIPS186-4): (A6730)

Table 7: Security Function Implementations

2.7 Algorithm Specific Information

SHA-1:

Digital signature generation using SHA-1 is non-approved and not allowed in approved services.

RSA:

For RSA signature verification, the module supports sizes 2048, 3072, and 4096 bits. Additionally, the module supports a modulus size of 1024 and 1536 bits for RSA signature verification. All supported modulus sizes have been CAVP tested and are conformant to FIPS 140-3 IG C.F.

Legacy Use:

In compliance with FIPS 140-3 IG C.K, the digital signature algorithm implementations have been CAVP tested against FIPS 186-5 where possible.

FIPS 186-2 CAVP testing was performed for RSA signature verification with a 1536-bit modulus. FIPS 186-4 CAVP testing was performed for digital signature verification using SHA-1 and RSA signature verification with a 1024-bit modulus. These algorithms are allowed for legacy use only.

2.8 RBG and Entropy

Cert Number	Vendor Name
E173	AMD

Table 8: Entropy Certificates

Name	Type	Operational Environment	Sample Size	Entropy per Sample	Conditioning Component
AMD TRNG Entropy Source	Physical	AMD Ryzen RIOD2.0	128	128	AES-CBC-MAC (A5337)

Table 9: Entropy Sources

The module provides an SP800-90Ar1-compliant Deterministic Random Bit Generator (DRBG) using CTR_DRBG mechanism with AES-256 for generation of key components of asymmetric keys, and random number generation. The module complies with the Public Use Document for ESV certificate E173 by reading entropy data from the 2048-bit FIFO, which corresponds to the GetEntropy() function. This function outputs 128 bits of entropy. The module constructs the 384-bit entropy input for the DRBG by requesting GetEntropy() three times and concatenating the result.

The DRBG does not employ a derivation function, does not support a personalization string, and does not support additional input. Consequently, the 384-bit entropy input is used directly as the DRBG seed, for both seeding and reseeding.

The operational environment on the ESV certificate is identical to the IOD in the Ryzen SoC, in which the sub-chip components are contained. Thus, the module is compliant with scenario 1 of IG 9.3.A. There are no maintenance requirements for the entropy source.

2.9 Key Generation

The module does not implement a key generation service, it only includes a key derivation service. Specifically, the module implements an SP 800-108r1 compliant KBKDF, using the HMAC SHA-256 PRF and a 32-bit counter. This implementation can be used to derive secret keys when provided with a pre-existing key-derivation key.

The resulting SSPs can be stored by the module in the Key Storage Block (if specified by the operator) or output as an API output parameter.

2.10 Key Establishment

The module does not implement any automated key establishment methods.

2.11 Industry Protocols

The module does not implement any industry protocol.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
SRAM	Data Input	API input parameters for data.
SRAM	Data Output	API output parameters for data.
SRAM	Control Input	API function calls, API input parameters for control.
SRAM	Status Output	API return codes, status values.
Power port	Power	Power port or pin on the SoC.

Table 10: Ports and Interfaces

The logical interfaces are logically separated from each other by the API design. The module does not implement a control output interface. The power interface is physically separated from any other interface.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not implement any authentication methods.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 11: Roles

No support is provided for multiple concurrent operators.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
RL_ARCL_Sha	Generate a (extendable output) message digest	2	Message, output length (XOF)	Message digest	Message digest XOF	Crypto Officer
RL_ARCL_Aes	Perform an AES operation (encrypt/decrypt)	2	Plaintext/ciphertext, AES key, IV (if applicable)	Plaintext/ciphertext	Encryption Decryption	Crypto Officer - AES key: W,E
RL_ARCL_Mac	Generate a MAC tag	2	Message, AES/HMAC key	MAC tag	MAC	Crypto Officer - AES key: W,E - HMAC key: W,E
RL_ARCL_Verify	Verify a message signature	2	Message, hash algorithm, signature, public key	Pass/fail	Signature verification Signature verification (legacy)	Crypto Officer - RSA public key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
RL_ARCL_DeriveKeyUsingPRF	Derive a key using SP 800-108r1 KDF	2	Key-derivation-key, derived key length	Derived key	Key derivation	Crypto Officer - Key-derivation key: W,E - Derived key: G,R
RL_ARCL_GenerateRandom	Generate random bytes	2	Output length	Random bytes	Random number generation	Crypto Officer - Entropy input: G,E,Z - DRBG seed: G,E,Z - Internal state (V, Key): G,W,E
RL_ARCL_FwImageLoadValidateWithKey	Verify the signature of a firmware image using a provided key	2	Firmware image, public key	Pass/fail	Signature verification	Crypto Officer - RSA public key: W,E
RL_ARCL_FwImageLoadValidate	Verify the signature of a firmware image using an embedded key	2	Firmware image	Pass/fail	Signature verification	Crypto Officer - RSA public key: W,E
RL_ARCL_KeyDbInstall	Verify the signature of a key	2	Key database image	Pass/fail	Signature	Crypto Officer - RSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	database image using an embedded key				verification	public key: W,E
RL_ARCL_KeyImageValidate	Verify the signature of a key image using an embedded key	2	Key image	Pass/fail	Signature verification	Crypto Officer - RSA public key: W,E
RL_ARCL_SelfTest	Perform on-demand self-tests	2	None	Pass/fail	None	Crypto Officer
RL_ARCL_RtlDeobfuscate	De-obfuscate some data using the RTL key	2	Obfuscated input data	De-obfuscated output data	None	Crypto Officer
RL_ARCL_Reconfig	Update the ASP register base address	None	Register base address	None	None	Crypto Officer
RL_ARCL_GetState (Show Status / Show Version)	Show the module status, version, and service indicator	None	None	Module status, version, service indicator	None	Crypto Officer
RL_ARCL_Scrap	Zeroize the KSB and prepare the module for end-of-life	None	None	None	None	Crypto Officer - AES key: Z - HMAC key: Z - Key-derivation key: Z - Derived key: Z - RSA

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						public key: Z
RL_ARCL_Shutdown	Zeroize the KSB and shut down the module	None	None	None	None	Crypto Officer - AES key: Z - HMAC key: Z - Key-derivation key: Z - Derived key: Z - RSA public key: Z
RL_ARCL_KeyDbRetire	Disable the installed key database image	None	None	None	None	Crypto Officer
RL_ARCL_ReinitHw	Reinitialize CCP hardware	None	None	None	None	Crypto Officer
RL_ARCL_GetShaInfo	Get SHA IV, message block size, and output hash length	None	SHA type	IV, message block size, output hash length	None	Crypto Officer
RL_ARCL_ModExp	Perform a modular exponentiation	None	Base, exponent, modulus	Result	None	Crypto Officer
RL_ARCL_RtlDisableKeyUsage	Disable usage of the RTL key	None	None	None	None	Crypto Officer
RL_ARCL_AddAddressMap	Register a new device address map	None	Device address map	None	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
RL_ARCL_GetRuntimeProfile	Get the runtime profile address	None	None	Runtime profile address	None	Crypto Officer
RL_ARCL_GetReadOnlyRegions	Get list of read-only regions	None	None	List of read-only regions	None	Crypto Officer
RL_ARCL_CcpDma	Copy data from a source to a destination using the CCP	None	SRAM address or KSB slot handle	SRAM address or KSB slot handle	None	Crypto Officer - AES key: R,W - HMAC key: R,W - Key-derivation key: R,W - Derived key: R - RSA public key: R,W
RL_ARCL_KsbAlloc	Allocate a slot in the KSB	None	Length, allocation type	KSB slot handle	None	Crypto Officer
RL_ARCL_KsbChangeUsage	Change attributes for a KSB slot	None	KSB slot, attributes	None	None	Crypto Officer
RL_ARCL_KsbGetAttributes	Retrieve attributes for a KSB slot	None	KSB slot	Attributes	None	Crypto Officer
RL_ARCL_KsbClear	Set the first 64 bytes of a KSB slot to zero	None	KSB slot	None	None	Crypto Officer
RL_ARCL_KsbLock	Lock a KSB slot	None	KSB slot	None	None	Crypto Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
RL_ARCL_KsbFree	Free and zeroize a previously allocated KSB slot	None	KSB slot	None	None	Crypto Officer - AES key: Z - HMAC key: Z - Key-derivation key: Z - Derived key: Z - RSA public key: Z
RL_ARCL_ZlibDecompress	Decompress zlib data	None	Compressed data	Uncompressed data	None	Crypto Officer
RL_ARCL_ClearInterrupt	Clear CCP interrupt for the flags	None	Flags	None	None	Crypto Officer
RL_ARCL_GetInterruptState	Check if CCP interrupt is signaled for the flags	None	Flags	Interrupt state	None	Crypto Officer
RL_ARCL_EnableInterrupt	Enable CCP interrupt for the flags	None	Flags	None	None	Crypto Officer
RL_ARCL_GetKeyUsageHistory	Check key usage since boot time	None	None	Key usage counters	None	Crypto Officer
RL_ARCL_RngReinit	Reinitialize the Entropy Source and DRBG	None	None	None	Random number generation	Crypto Officer - Entropy input: G,E,Z - DRBG seed: G,E,Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						- Internal state (V, Key): G
RL_ARCL_RngReseed	Reseed the DRBG	None	None	None	Random number generation	Crypto Officer - Entropy input: G,E,Z - DRBG seed: G,E,Z - Internal state (V, Key): G,W,E
RL_ARCL_DeInitVcq	Disable and clear the virtual queue VCQ0	None	None	None	None	Crypto Officer

Table 12: Approved Services

For the above table, the convention below applies when specifying the access permissions (type) that the service has for each SSP.

- **Generate (G)**: The module generates or derives the SSP.
- **Read (R)**: The SSP is read from the module (e.g., the SSP is output).
- **Write (W)**: The SSP is updated, imported, or written to the module.
- **Execute (E)**: The module uses the SSP in performing a cryptographic operation.
- **Zeroize (Z)**: The module zeroizes the SSP.
- **N/A**: The module does not access any SSP or key during its operation.

The module provides three different API layers, each with distinct services:

1. The HAL layer, which is the low-level interface for CCP and SIB hardware.
    a. The HAL CCP API provides non-approved cryptographic functionality.
    b. The HAL SIB API provides functionality to interact with the Key Storage block, DRBG, and Security State.

2. The ARCL layer, which provides high-level cryptographic (both approved and non-approved) and non-cryptographic functionality.
3. The Firmware image processing layer, which is implemented on top of the ARCL API and provides high-level functionalities, such as firmware image validation and key database management.

The ARCL API layer provides the RL_ARCL_GetState function which returns the ArclState, ArclVersion, and FipsIndicatorStatus values:

- The ArclState value serves as the module's status indicator and is used to indicate the error states.
- The ArclVersion value contains the module's versioning information.
- The FipsIndicatorStatus value services as the approved service indicator. If this value is set to 2, the previously requested ARCL service was approved. If this value is set to any other value, the service was not approved. The HAL CCP API and HAL SIB API only provide non-approved services.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
RL_ARCL_Mac	Generate a MAC tag	HMAC with key lengths less than 112 bits	Crypto Officer
RL_ARCL_RsaPssVerifySignature	Verify a pre-hashed message signature	RSA (pre-hashed message)	Crypto Officer
RL_ARCL_MeasureMemoryPerPcr	PCR-based memory measurement	SHA-384 with non-standard initial hash value	Crypto Officer
CCP_HAL API	Any API in the CCP_HAL API layer	CCP_HAL algorithm	Crypto Officer
SIB_HAL API	Any API in the SIB_HAL API layer	SIB_HAL algorithm	Crypto Officer

Table 13: Non-Approved Services

4.5 External Software/Firmware Loaded

Upon startup, the libROM firmware component loads the overlay firmware from external storage (SPI flash) into the sub-chip cryptographic subsystem. The integrity of the overlay firmware is determined by verifying an RSA-PSS 4096 with SHA-384 signature stored in the overlay firmware that was computed at build time. If the signature verification fails, the overlay firmware load test fails. The public key used to verify this signature is stored inside the libROM firmware component of the module, the private key associated with this public key is controlled by the vendor.

All data output is inhibited during the execution of the overlay firmware load test and the overlay firmware loading process.

5 Software/Firmware Security

5.1 Integrity Techniques

The integrity of the libROM component of the module is verified by comparing a SHA-384 digest value calculated at runtime with the SHA-384 digest value stored in the module that was computed at build time.

The integrity of the overlay firmware component of the module is discussed in Section 4.5.

5.2 Initiate on Demand

The module provides the RL_ARCL_SelfTest service to perform self-tests on demand. Among those self-tests is the integrity test, as part of the pre-operational self-tests. More details on the API are provided by the vendor in its developer’s manual.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Limited

How Requirements are Satisfied:

Any SSPs contained within the module are protected by the hardware and firmware restrictions implemented by the Key Storage Block. Only the module has access to these SSPs, and access is only possible through the defined interfaces.

6.2 Configuration Settings and Restrictions

No configuration of the operational environment is required for the module to operate in an approved mode. Therefore, there are no rules, settings, or restrictions to the configuration of the operational environment.

7 Physical Security

7.1 Mechanisms and Actions Required

Mechanism	Inspection Frequency	Inspection Guidance
Opaque sealing coat	No actions are required to maintain the physical security of the module	No actions are required to maintain the physical security of the module

Table 14: Mechanisms and Actions Required

The module provides no additional physical security techniques.

8 Non-Invasive Security

The module does not implement any non-invasive security mechanisms.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
Hardware registers	Hardware registers store the SSPs used by the hardware DRBG	Dynamic
Key Storage Block (KSB)	Hardware block used to securely store SSPs while the module is operational	Dynamic
SRAM	Temporary storage for SSPs used by the module as part of service execution	Dynamic

Table 15: Storage Areas

The Key Storage Block maintains internal separation of the SSPs (including CSPs) in the approved and non-approved modes of operation using a “virtual queue” mechanism: virtual queue 0 is exclusively used for approved services, whereas virtual queue 1 is always used for non-approved services. The HAL SIB API provides functionality to interact with the Key Storage Block for queue 1 only.

The module does not perform persistent storage of SSPs; SSPs in use by the module exist in volatile memory only.

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API input parameters	Operator calling application (TOEPP)	Cryptographic module	Plaintext	Manual	Electronic	
API output parameters	Cryptographic module	Operator calling application (TOEPP)	Plaintext	Manual	Electronic	

Table 16: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
RL_ARCL_KsbFree	Zeroize a single KSB slot	Memory occupied by the SSP is overwritten with zeroes, which renders the SSP value irretrievable. Completion of the function indicates that the zeroization procedure succeeded.	By calling the RL_ARCL_KsbFree function
RL_ARCL_Shutdown	Zeroize all data stored in the KSB	Memory occupied by the SSPs is overwritten with zeroes, which renders the SSP values irretrievable. Completion of the function indicates that the	By calling the RL_ARCL_Shutdown function

Zeroization Method	Description	Rationale	Operator Initiation
		zeroization procedure succeeded.	
RL_ARCL_Scrap	Zeroize all data stored in the KSB	Memory occupied by the SSPs is overwritten with zeroes, which renders the SSP values irretrievable. Completion of the function indicates that the zeroization procedure succeeded.	By calling the RL_ARCL_Scrap function
Remove power from the SoC	De-allocates the volatile memory used to store SSPs	Volatile memory used by the module is overwritten within nanoseconds when power is removed	By removing power
Automatic	Automatically zeroized by the module when no longer needed	Every service overwrites its temporary memory upon completion, which renders any SSP values used by the service irretrievable. Completion of the service indicates that the zeroization procedure succeeded.	N/A

Table 17: SSP Zeroization Methods

All data output is inhibited during zeroization.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES key	Symmetric key used for AES operations	128, 192, 256 bits - 128, 192, 256 bits	Symmetric - CSP			Encryption Decryption MAC
HMAC key	Symmetric key used for HMAC operations	112-256 bits - 112-256 bits	Symmetric - CSP			MAC
Key-derivation key	Symmetric key used to derive other symmetric keys	112-256 bits - 112-256 bits	Symmetric - CSP			Key derivation
Derived key	Symmetric key derived from a key-derivation key	112-256 bits - 112-256 bits	Symmetric - CSP	Key derivation		
Entropy input	Entropy input used to seed the DRBG	384 bits - 384 bits	Entropy input - CSP	Random number generation		Random number generation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG seed	DRBG seed derived from entropy input	384 bits - 256 bits	DRBG seed - CSP	Random number generation		Random number generation
Internal state (V, Key)	Internal state of the CTR_DRBG instance	384 bits - 256 bits	Internal state - CSP	Random number generation		Random number generation
RSA public key	Public key used for RSA	1024, 1536, 2048, 3072, 4096 bits - 80, 96, 112, 128, 150 bits	Public key - PSP			Signature verification

Table 18: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES key	API input parameters API output parameters	Key Storage Block (KSB):Plaintext SRAM:Plaintext	KSB: until explicitly removed or the module ends its operation; SRAM: for the duration of the service	RL_ARCL_KsbFree RL_ARCL_Shutdown RL_ARCL_Scrap Remove power from the SoC Automatic	
HMAC key	API input parameters API output parameters	Key Storage Block (KSB):Plaintext SRAM:Plaintext	KSB: until explicitly removed or the module ends its operation; SRAM: for the duration of the service	RL_ARCL_KsbFree RL_ARCL_Shutdown RL_ARCL_Scrap Remove power from the SoC Automatic	
Key-derivation key	API input parameters API output parameters	Key Storage Block (KSB):Plaintext SRAM:Plaintext	KSB: until explicitly removed or the module ends its operation; SRAM: for the duration of the service	RL_ARCL_KsbFree RL_ARCL_Shutdown RL_ARCL_Scrap Remove power from the SoC Automatic	
Derived key	API output parameters	Key Storage Block (KSB):Plaintext SRAM:Plaintext	KSB: until explicitly removed or the module ends its	RL_ARCL_KsbFree RL_ARCL_Shutdown RL_ARCL_Scrap Remove power from	Key-derivation key:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
			operation; SRAM: for the duration of the service	the SoC Automatic	
Entropy input		Hardware registers:Plaintext	From generation until DRBG seed is created	Remove power from the SoC	
DRBG seed		Hardware registers:Plaintext	While the DRBG is instantiated	Remove power from the SoC	Entropy input:Derived From
Internal state (V, Key)		Hardware registers:Plaintext	From DRBG instantiation until DRBG termination	Remove power from the SoC	DRBG seed:Derived From
RSA public key	API input parameters API output parameters	Key Storage Block (KSB):Plaintext SRAM:Plaintext	KSB: until explicitly removed or the module ends its operation; SRAM: for the duration of the service	RL_ARCL_KsbFree RL_ARCL_Shutdown RL_ARCL_Scrap Remove power from the SoC Automatic	

Table 19: SSP Table 2

9.5 Transitions

The SHA-1 algorithm as implemented by the module will be non-approved for all purposes, starting January 1, 2031.

10 Self-Tests

While the module is executing the self-tests, services are not available, and data output (via the data output interface) is inhibited until the tests are successfully completed. The module does not return control to the calling application until the tests are completed.

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
SHA2-384 (A6730)	N/A	Message digest	SW/FW Integrity	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Integrity test on the libROM firmware component at power up

Table 20: Pre-Operational Self-Tests

The pre-operational firmware integrity test on the libROM firmware component is performed automatically when the module is initialized. If this test fails, the module transitions to the hard error state.

10.2 Conditional Self-Tests

As part of the initialization, the libROM firmware component loads the overlay firmware component and performs the firmware load test on the overlay firmware. Only if this test succeeds, will the module move to the operational state. Similar to the pre-operational integrity test, if the firmware load test fails, the module transitions to the hard error state.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA SigVer (FIPS186-5) (A6730)	4096-bit key, SHA-384	Signature verification	SW/FW Load	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Firmware load test on the overlay firmware component	Power up
SHA-1 (A6730)	0-bit message	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Message digest	Prior to first approved use of SHA-1
SHA2-256 (A6730)	0-bit message	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Message digest	Prior to first approved use of SHA-224 or SHA-256

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
SHA2-512 (A6730)	0-bit message	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Message digest	Prior to libROM firmware integrity test
SHA3-512 (A6730)	0-bit message	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Message digest	Prior to first approved use of SHA-3 or SHAKE
AES-ECB (A6730)	128-bit key	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Encryption	Prior to first approved use of AES ECB, CBC, or CTR
AES-ECB (A6730)	128-bit key	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Decryption	Prior to first approved use of AES ECB, CBC, or CTR
AES-CMAC (A6730)	128-bit key	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	MAC tag generation	Prior to first approved use of AES CMAC
HMAC-SHA2-384 (A6730)	384-bit key, SHA-384	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	MAC tag generation	Prior to first approved use of HMAC
KDF SP800-108 (A6730)	256-bit key-derivation key, 128-bit derived key	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Key-based key derivation	Prior to first approved use of KBKDF
Entropy Source	Cutoff: 5 samples	RCT	CAST	Entropy Source is operational	SP 800-90B	Initialization of the

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
					start-up health test ran over 4096 samples	Entropy Source
Entropy Source	Cutoff: 16 samples	APT	CAST	Entropy Source is operational	SP 800-90B start-up health test ran over 4096 samples	Initialization of the Entropy Source
Entropy Source	Cutoff: 5 samples	RCT	CAST	Entropy Source produces entropy	SP 800-90B continuous health test	DRBG seeding
Entropy Source	Cutoff: 16	APT	CAST	Entropy Source produces entropy	SP 800-90B continuous health test	DRBG seeding
Counter DRBG (A7030)	AES-256	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	SP 800-90Ar1 (instantiate, reseed, generate) health test	Prior to first approved use of the CTR_DRBG
RSA SigVer (FIPS186-5) (A6730)	2048-bit key with SHA-384	KAT	CAST	ArclState is set to ARCL_STATE_VALIDATED_OPERATIONAL	Signature verification	Prior to overlay firmware load test

Table 21: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA2-384 (A6730)	Message digest	SW/FW Integrity	On demand	Manually

Table 22: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
RSA SigVer (FIPS186-5) (A6730)	Signature verification	SW/FW Load	On demand	Manually
SHA-1 (A6730)	KAT	CAST	On demand	Manually
SHA2-256 (A6730)	KAT	CAST	On demand	Manually
SHA2-512 (A6730)	KAT	CAST	On demand	Manually
SHA3-512 (A6730)	KAT	CAST	On demand	Manually
AES-ECB (A6730)	KAT	CAST	On demand	Manually
AES-ECB (A6730)	KAT	CAST	On demand	Manually
AES-CMAC (A6730)	KAT	CAST	On demand	Manually
HMAC-SHA2-384 (A6730)	KAT	CAST	On demand	Manually
KDF SP800-108 (A6730)	KAT	CAST	On demand	Manually
Entropy Source	RCT	CAST	On demand	Manually
Entropy Source	APT	CAST	On demand	Manually
Entropy Source	RCT	CAST	Every sample	Manually
Entropy Source	APT	CAST	Every sample	Manually
Counter DRBG (A7030)	KAT	CAST	On demand	Manually
RSA SigVer (FIPS186-5) (A6730)	KAT	CAST	On demand	Manually

Table 23: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Soft Error	The module only responds to status, zeroization, and self-test service requests	Cryptographic algorithm self-test error	Invoke RL_ARCL_SelfTest service	ArclState = 8
Hard Error	The module does not respond to any service requests and must be reset	FW integrity test error or FW load test error	Power off the module	ArclState = 16

Table 24: Error States

In the Soft Error state, the module outputs the error type through the status indicator and status output interface. Moreover, the data input and data output interfaces are inhibited, and the module only accepts control input.

In the Hard Error state, no input or output is possible at all.

10.5 Operator Initiation of Self-Tests

The operator can request on-demand self-tests by invoking the RL_ARCL_SelfTest service. This service executes all self-tests listed above.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

To detect any potential tampering during delivery of the module, the user can verify the Thermoform or JEDEC tray is securely strapped, and vacuum sealed in the moisture barrier bag. Additionally, the SoC itself provides tamper evidence as specified in Section 7.

Upon delivery, no further installation or configuration is required for the hardware to operate as the validated module in conformance with the rules in this Security Policy document. The module implicitly transitions between the approved mode and the non-approved mode when appropriate.

11.2 Administrator Guidance

All the functions, ports and logical interfaces described in this document are available to the Crypto Officer. The module implicitly transitions between the approved mode and the non-approved mode contingent on the service that is invoked. Therefore, there are no special procedures to administer the approved or non-approved modes.

11.3 Non-Administrator Guidance

The module implements only the Crypto Officer. There are no requirements for non-administrator operators.

11.4 Design and Rules

Not applicable.

11.5 Maintenance Requirements

Not applicable.

11.6 End of Life

The process for performing “End of Life” occurs at the chronological point of 10 years starting from manufacturing date of the module.

The module does not possess persistent storage of SSPs. The SSP value only exists in volatile memory and that value vanishes when the module is powered off. The procedure for secure sanitization of the module at the end of life is achieved in the following way:

- The Crypto Officer issues a call to RL_ARCL_Scrap which releases any resources held by the module, cleans up global workspace, and zeroizes any SSPs. Upon completion of this service, all SSPs are removed from the module, so that the module may either be distributed to other operators or disposed.

After sanitization, the Crypto Officer should blow the “scrap” (end-of-life) fuse to render the module non-operational.

12 Mitigation of Other Attacks

The module does not implement security mechanisms to mitigate other attacks.

A Glossary and Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
ARCL	AMD Root of Trust Crypto Library
ASP	AMD Secure Processor
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCP	Cryptographic Co-Processor
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
FIPS	Federal Information Processing Standards
GCTR	Galois Counter
HAL	Hardware Abstraction Layer
HMAC	Keyed-Hash Message Authentication Code
IAPM	Integrity-Aware Parallelizable Mode
IG	International Guidance
IV	Initialization Vector
JEDEC	Joint Electron Device Engineering Council
KAT	Known Answer Test
KSB	Key Storage Block
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
OFB	Output Feedback
OTP	One-Time Programmable
PCT	Pair-wise Consistency Test
PKI	Public Key Infrastructure
PSP	Public Security Parameter
PSS	Probabilistic Signature Scheme
ROM	Read-Only Memory

RSA	Rivest Shamir Adleman
RTL	Register-Transfer Level
SHA	Secure Hash Algorithm
SHAKE	Secure Hash Algorithm with Keccak
SIB	Security Infrastructure Block
SoC	System on Chip
SRAM	Static Random-Access Memory
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment's Physical Perimeter
TRNG	True Random Number Generator
XOF	Extendable Output Function
XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

B References

FIPS 140-3	**FIPS PUB 140-3 - Security Requirements For Cryptographic Modules** March 2019 https://doi.org/10.6028/NIST.FIPS.140-3
FIPS 140-3 IG	**Implementation Guidance for FIPS PUB 140-3 and the Cryptographic Module Validation Program** https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf
FIPS 180-4	**Secure Hash Standard (SHS)** August 2015 https://doi.org/10.6028/NIST.FIPS.180-4
FIPS 186-2	**Digital Signature Standard (DSS)** January 2000 https://csrc.nist.gov/files/pubs/fips/186-2/final/docs/fips186-2.pdf
FIPS 186-4	**Digital Signature Standard (DSS)** July 2013 https://doi.org/10.6028/NIST.FIPS.186-4
FIPS 186-5	**Digital Signature Standard (DSS)** February 2023 https://doi.org/10.6028/NIST.FIPS.186-5
FIPS 197	**Advanced Encryption Standard (AES)** November 2001; Updated May 2023 https://doi.org/10.6028/NIST.FIPS.197-upd1
FIPS 198-1	**The Keyed-Hash Message Authentication Code (HMAC)** July 2008 https://doi.org/10.6028/NIST.FIPS.198-1
FIPS 202	**SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions** August 2015 https://doi.org/10.6028/NIST.FIPS.202
SP 800-38A	**Recommendation for Block Cipher Modes of Operation: Methods and Techniques** December 2001 https://doi.org/10.6028/NIST.SP.800-38A
SP 800-38B	**Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication** May 2005; Updated October 2016 https://doi.org/10.6028/NIST.SP.800-38B
SP 800-38D	**Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 https://doi.org/10.6028/NIST.SP.800-38D
SP 800-38E	**Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices** January 2010 https://doi.org/10.6028/NIST.SP.800-38E

SP 800-90Ar1	**Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 https://doi.org/10.6028/NIST.SP.800-90Ar1
SP 800-90B	**Recommendation for the Entropy Sources Used for Random Bit Generation** January 2018 https://doi.org/10.6028/NIST.SP.800-90B
SP 800-108r1	**Recommendation for Key Derivation Using Pseudorandom Functions** August 2022; Updated February 2024 https://doi.org/10.6028/NIST.SP.800-108r1-upd1
SP 800-131Ar2	**Transitioning the Use of Cryptographic Algorithms and Key Lengths** March 2019 https://doi.org/10.6028/NIST.SP.800-131Ar2
SP 800-133r2	**Recommendation for Cryptographic Key Generation** June 2020 https://doi.org/10.6028/NIST.SP.800-133r2
SP 800-140Br1	**Cryptographic Module Validation Program (CMVP) Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B** November 2023 https://doi.org/10.6028/NIST.SP.800-140Br1