Silver Spring Networks, Inc.

Endpoint Security Module

FIPS 140-2 Non-Proprietary Security Policy Version 1.0

Silver Spring Networks, Inc.
230 W Tasman Dr
San Jose, CA 95134
3/20/2019

Table of Contents

List of Tables

1. Introduction

The Silver Spring Networks, Inc. End Point Security Module (SSN ESM) is a cryptographic processing unit that was designed from the ground up for Federal Information Processing Standard (FIPS) 140-2 level 3 security isolation. The security module provides FIPS-approved cryptographic algorithms and employs a FIPS 140-2 level 3 isolation boundary at a sub-chip boundary inside a system on chip (SoC)– the Silver Spring Networks, Inc. V5 chip. The SSN ESM is based on the HardCache licensed technology.

Module name	**Firmware Version**	**Hardware Version**
Endpoint Security Module	ROM version 82136 with Applet version 1.0.0	130-0117-01.ESM

Table 1 - Module Information and Versioning

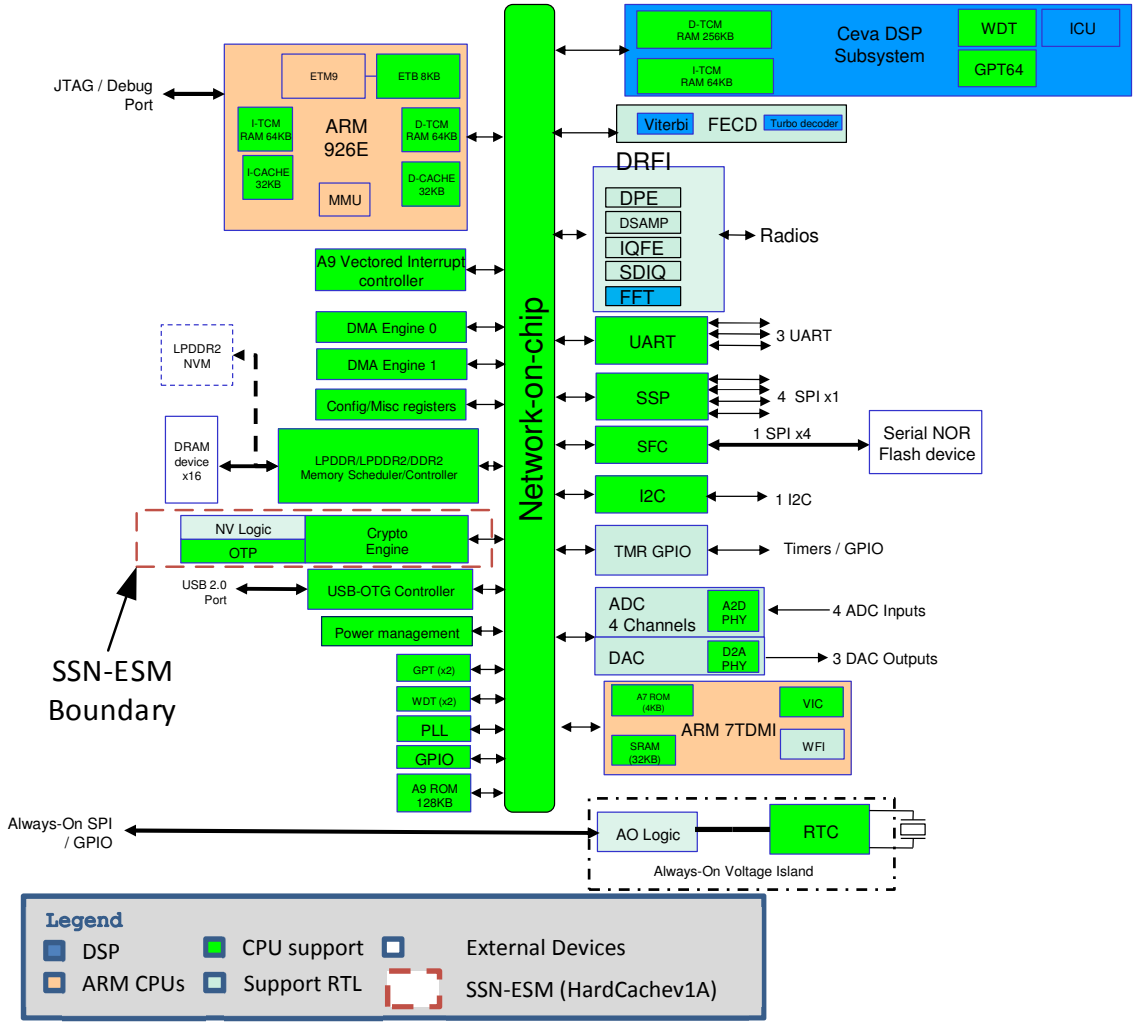

Figure 1 - Silver Spring Networks, Inc. V5 System on Chip

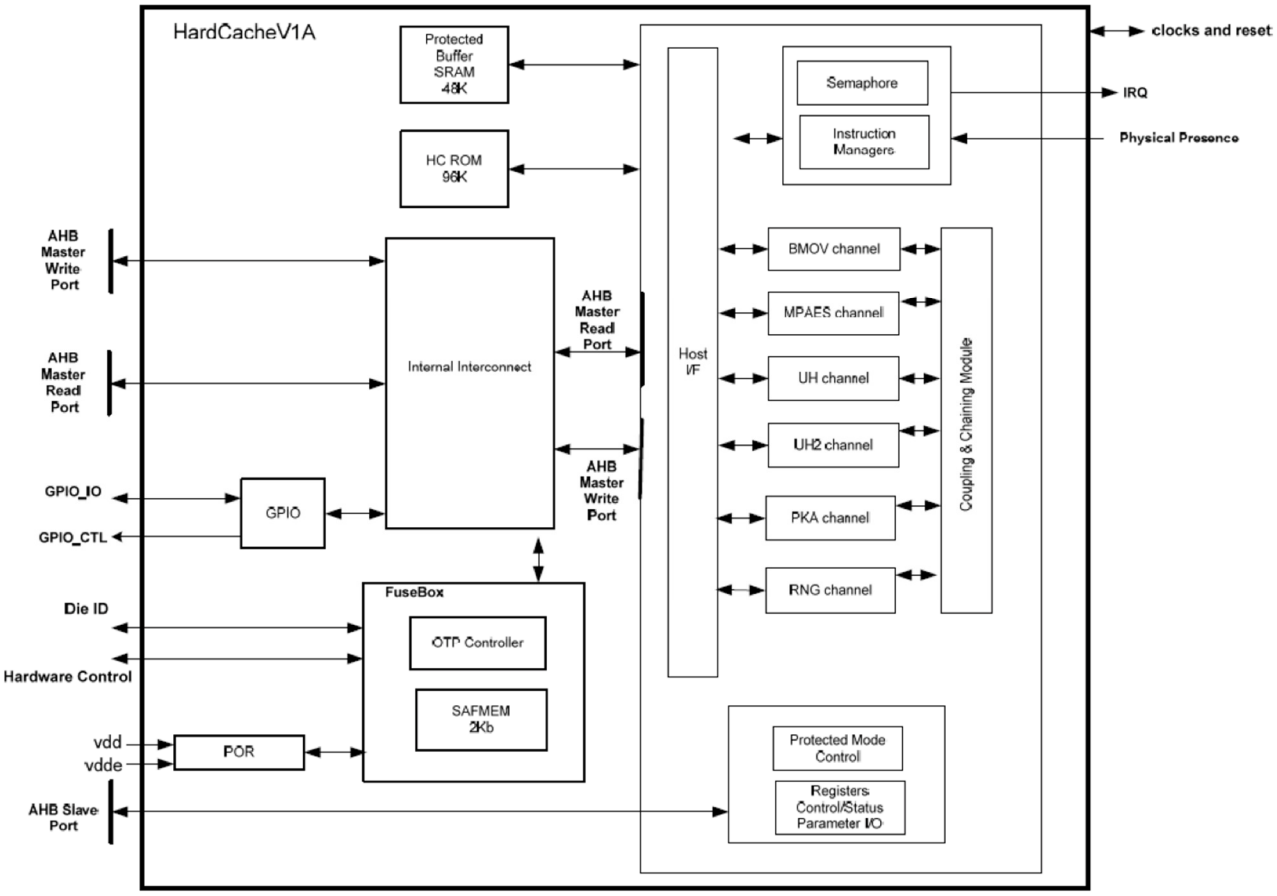

Figure 2–SSN ESM/HardCache block diagram
sub-chip cryptographic subsystem boundary hardware instantiation

2. Physical Boundary

The SSN ESM is a subcomponent or sub-chip of a custom ASIC. Per FIPS 140-2 IG, Section 1.20:

> *"For a hardware module, the minimum defined physical boundary in FIPS 140-2 is a single-chip."*

For the purposes of FIPS 140-2 evaluation, the physical boundary for the SSN ESM is the outer layer of the custom ASIC shown below.

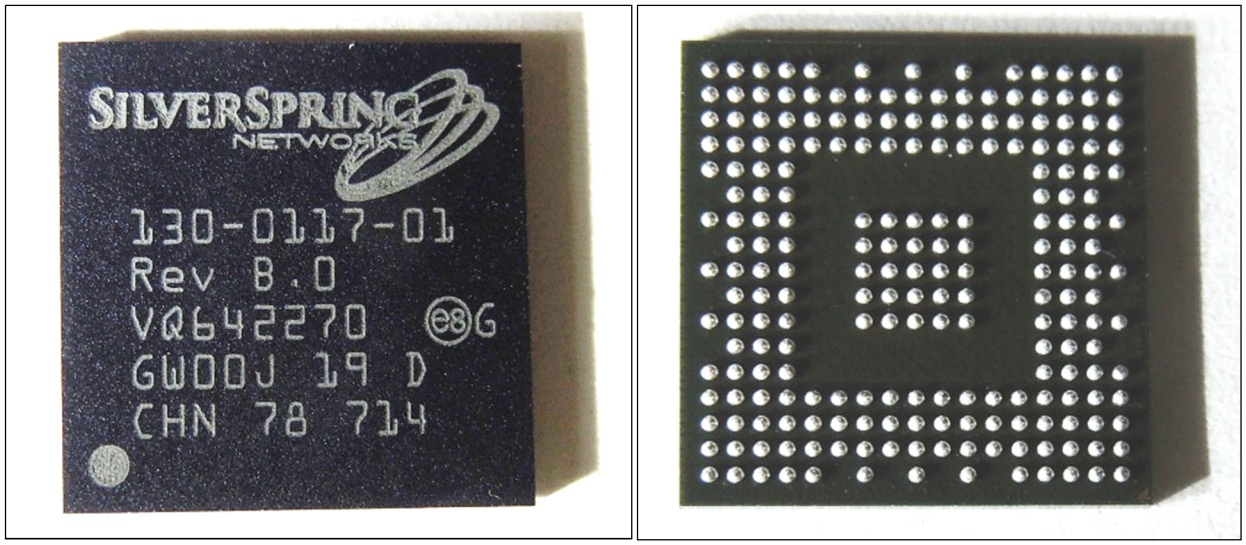

Figure 3*– Silver Spring Networks, Inc. V5 SoC containing the SSN ESM Physical boundary*

3. Sub-Chip Cryptographic Subsystem Boundary (Logical Boundary).

The SSN ESM Sub-Chip Cryptographic Subsystem Boundary (SCCSB) consist of two parts: The set of circuitry cores and the associated firmware.

3.1. Circuitry Cores

Hardware instantiation[1] of the SSN ESM contains the following included elements (See Figure 2–SSN ESM/HardCache block diagram
sub-chip cryptographic subsystem boundary above):

- **Cryptographic Accelerators for symmetric and asymmetric cryptographic operations**
- **Instruction processing units**
- **Random-access memory (RAM)**
- **Read-only memory (ROM)**
- **Registers**
- **Interface logic blocks**
- **Hardware random number generator**
- **Hardware FSM (Fusebox) implementing a small one-time programmable (OTP) memory area within the hardware cryptographic boundary**

The Silver Spring Networks, Inc. V5 SoC containing the SSN ESM includes the following elements excluded from FIPS 140-2 requirements and this validation (See Figure 1 - Silver Spring Networks, Inc. V5 System on Chip above):

- **ARM 9 processor**
- **ARM 7 processor**
- **Digital signal processor**
- **Radio front ends, Downstream Radio Frequency Interface (DRFI)**
- **Digital-to-analog (D/A) converter and analog-to-digital (A/D) converters**
- **DMA engines**
- **Interrupt controllers**
- **General-purpose inputs/outputs (GPIOs)**
- **Timers**
- **Real time clock**
- **SPI and I2C interfaces**
- **Serial flash controller**
- **Memory management units**
- **ROM memory**
- **USB Controller**
- **Internal Bus Matrix/Control**

[1] FIPS 140-2 IG, Section 1.20 refers to hardware instantiation as "…the set of hard and/or soft circuitry cores … which represents a sub-chip cryptographic subsystem boundary of a single-chip hardware module."

- **DDR Controller/Bus**
- **UART**
- **Thermal sensor**
- **Debug interface**

3.2. Logical Boundary

The logical boundary includes everything physically within the SSN ESM circuitry core as well as the firmware for the ESM. In addition, a number of CSPs, public keys, and public data have been sealed or wrapped by keys that never leave the SSN ESM, and that can be considered logically and cryptographically within the ESM, even if their at-rest location is external to the physical boundaries of the ESM.

Wrapped secret and private keys consist of keys generated on behalf of the user and wrapped (using the AES key wrap mode) and exported from the module when not being used. The key used to wrap the exported key is generated within the SSN ESM and never leaves that boundary. The typical key used for this purpose is the "Instance Seal Key."

Sealed public keys and data consist of data that has been signed using a keyed-hash message authentication code (HMAC) key specific to either the module type (data sealed during manufacturing such as module firmware) or to the specific module (the public keys used as part of the login sequence). The typical key used for this is the "Module Seal Key."

For the wrapped data, if the key within the module used to wrap the data is zeroized, the wrapped data blob becomes useless and cryptographically uninteresting.

4. Cryptographic Algorithms

The cryptographic module supports the following Approved cryptographic algorithms:

CAVP Cert	Algorithm	Standard	Mode/Method	Key lengths, curves or mod-uli	Usage
5101	AES	FIPS 197, SP 800-38A	ECB,CBC, CFB8, OFB	128, 192, 256	Data Encryption/ Decryption
5238	AES	FIPS 197, SP 800-38F	Key Wrap	256	Key Wrapping /Unwrapping
1709	CVL	SP 800-56A	ECC CDH Primitive	P-256, P-384	Shared Secret Computation

CAVP Cert	Algorithm	Standard	Mode/Method	Key lengths, curves or moduli	Usage
1709	CVL	SP 800-56A	FFC DH Primitive	(2048, 256) Schemes: dhHybrid1 dhEphem dhHybrid1Flow dhOneFlow dhStatic	Shared Secret Computation
1359	DSA	FIPS 186-4	Key Pair Generation	(2048, 256)	Prerequisite to KAS FFC Only
1906	DRBG	SP 800-90A Revision 1	HMAC-DRBG SHA-512		Deterministic Random Bit Generation
1362	ECDSA	FIPS 186-4	PKG, SigGen, Sigver	P-256, P-384	Digital Signature Generation and Verification, Key Pair Generation
3402	HMAC	FIPS 198-1	HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	112 bits or greater for SHA1;128 bits or greater for SHA224 and SHA256; 192 bits or greater for SHA384; 256 bits or greater for SHA512	Message Authentication
164	KAS ECC	SP 800-56A	Ephemeral Unified	P-256	Key Agreement
178	KBKDF	SP800-108	Counter using HMAC-SHA256	Per HMAC-SHA256 requirements	Key Derivation
1710	CVL KDF[2]	SP800-135	IKEv1, IKEv2, TLSv1, TLSv2, ANS X9.63		Key Derivation

[2] As per FIPS 140-2 IG D.11, no parts of these protocols, other than the KDF, have been tested by the CAVP and CMVP.

CAVP Cert	Algorithm	Standard	Mode/Method	Key lengths, curves or mod-uli	Usage
2800	RSA	FIPS 186-4	RSASSA-PKCS1_V1_5, RSASSA-PSS	1024(verify only), 2048	Digital Signature Generation and Verification, Key Pair Generation
1711	CVL RSA	FIPS186-4 RSA; PKCS#1 v2.1	RSA-SP1	2048	Signature Primitive
4148	SHS	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	N/A	Message Digest

Table 2- Approved algorithms

The cryptographic module supports the following non-Approved but allowed algorithms:

Algorithm	Caveat	Use
NDRNG	Not user accessible	Seeding for the DRBG

Table 3 - Non-approved algorithms

5. Physical Ports and Logical Interfaces

All control ports to the Cryptographic module are internal to the SoC. No physical connections exist between the external (SoC electrical connection pins) and internal module except for power and data. All control access to the Cryptographic Module must be made through internal (sub-chip) ports from the SoC CPUs. Therefore, it is physically impossible to control the Cryptographic module from the physical pins of the device except by indirect power disruption.

The cryptographic module supports physical connections through chip-level traces grouped as follows:

Physical port	Logical interface
Power is supplied by power planes within the IC.	Power
Control I/O is limited to two memory-mapped regions of registers. Each region has one interrupt port used to signal asynchronous status from the module.	Control I/O
Physical Presence. A GPIO pin is used to signal to the module physical presence to enable certain operationally sensitive functions.	Control I/O (Input only)

Physical port	Logical interface
Data input is initiated and validated by the cryptographic module, which retrieves data using DMA. This indirectly gives access to the data bus of external memory; however, the cryptographic unit does not allow transactions to be initiated on this bus.	Data Input
Data output is initiated and validated by the cryptographic module using DMA.	Data Output
Interrupt Port: A single-bit interrupt port is used to signal a change in status of the cryptographic module. This port is connected to the interrupt controller of the containing SoC.	Status Output
Status Register: The Status Register provides indications for Run state; Bus Error; Reset; Channel Error; Events; OTP State; Physical Presence; Command Type.	Status Output

Table 4 - Cryptographic boundary ports

6. Security Rules

The following specifies the security rules under which the cryptographic module shall operate:

- The cryptographic module provides a single, post-manufacturing mode of operation that is FIPS approved.
- Approved cryptographic algorithms shall be used to protect sensitive unclassified data. The cryptographic module shall not implement any Non-Approved cryptographic algorithms with the exception of a hardware non-deterministic random number generator for seeding the DRBG.
- The cryptographic module design shall correspond to its finite state model and design specifications.
- Critical security parameters shall be protected against unauthorized disclosure, modification, and substitution. Public keys used in support of module security shall be protected against unauthorized modification and substitution.
- The cryptographic modules shall not allow the input or output of plaintext critical security parameters.
- Logical separation between the data inputs, control inputs, data outputs, and status outputs shall be enforced. The module shall receive power from chip traces designated for that purpose.

- While in an error state, the cryptographic module shall inhibit all data outputs, cease to provide cryptographic services, and shall provide an error status indication. This error status is reported in a status register and requires a complete reset and re-login in the event of an error status. The module enforces a zero tolerance for errors.
- All data outputs shall be inhibited during self-test conditions. Status information used to provide an indication of the results of self-tests shall not contain any CSPs, plaintext data, or other information that could be used to compromise the cryptographic module.
- All data output paths shall be logically disconnected from the processes performing key generation and zeroization.
- The cryptographic module shall not support manual private or secret key entry or split-knowledge processes.
- Only the User role shall be supported for general cryptographic operations. Only functions that would logically be performed by a Cryptographic officer can be controlled by the Cryptographic officer, and all user-specific cryptographic functions (such as signing, encryption using sensitive keys associated with the User role) are locked out.
- Identity-based authentication shall be enforced for all services that utilize CSPs, utilize approved security functions, and are otherwise security relevant.
- Bypass capabilities, bypass states, and bypass tests shall not be supported.
- Services inputs shall consist of all data or control inputs to the module that initiate or obtain specific services, operations, or functions. Services outputs shall consist of all data and status outputs that result from the services, operations, or functions initiated or obtained by services inputs.
- The feedback mechanism used during the authentication process shall not disclose any CSP or other information that could be used to compromise the cryptographic module.
- Physical security is provided by the cryptographic module through a hard enclosure that prevents penetration to the depth of the underlying circuitry without having a high probability that the module is damaged to the extent that it no longer functions. The cryptographic module is constructed of production-grade materials with standard passivation and tamper evidence, and is opaque within the visible spectrum.
- The cryptographic module shall include a non-modifiable operational environment. It shall not be possible to physically or logically alter the executable instructions and logic that reside within the cryptographic boundary.
- The module supports at minimum 256-bits of entropy. Cryptographic keys shall be generated using the implemented Approved DRBG. The key generation process shall provide

an equivalent computational resistance to attack of at least 256 bits of security. Intermediate key generation values shall not be output from the cryptographic module during the key generation process.

- Sensitive cryptographic keys shall be entered to and output from the cryptographic boundary in an encrypted form.
- A key-to-entity association shall be enforced for all keys that are entered into or output from and stored within the cryptographic boundary.
- The implemented zeroization techniques shall be performed in a time that is not sufficient to compromise plaintext secret and private keys and CSPs.

- **Devices which contain the SSN ESM shall conform to the appropriate EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B: Unintentional Radiators. See section 12 below. It is the responsibility of the integrator to accomplish any testing or certification necessary to meet these requirements.**

- The cryptographic module shall support error states associated with each self-test and output the expected error indicator upon entering into an error state resulting from a failed self-test.

    - The following power-up self-tests shall be supported:

        1. Firmware integrity tests:

            - 128-bit EDC ROM integrity test
            - ECDSA P-256 SHA-256 applet integrity test

        2. Cryptographic algorithm tests:

            - HMAC-SHA-1 KAT
            - HMAC-SHA-256 KAT
            - HMAC-SHA-512-KAT
            - AES-256-CBC Encrypt KAT
            - AES-256-CBC Decrypt KAT
            - SP 800-38F AES Key Wrap KAT
            - SP 800-38F AES Key Unwrap KAT
            - ECDSA P-256 SHA-256 Sign KAT
            - ECDSA P-256 SHA-256 Verify KAT
            - ECC CDH P-256 Primitive “Z” Computation KAT
            - DRBG (HMAC-SHA-512) KAT
            - SHA-1 KAT
            - RSA 2048 SHA-256 Sign KAT
            - RSA 2048 SHA-256 Verify KAT
            - SP 800-135 IKEv1 KDF KAT
            - SP 800-135 IKEv2 KDF KAT
            - SP 800-135 TLSv1.0/1.1 KDF KAT
            - SP 800-135 TLSv1.2 KDF KAT
            - SP 800-108 KBKDF KAT
            - SP 800-135 ANSI X9.63 KDF KAT
            - SP 800-56A Single-step KDF KAT
            - FFC DH 2048 Primitive “Z” Computation KAT

3. Critical functions tests:
    - RAM test
    - OTP test

- The following conditional self-tests shall be supported:
    - TRNG continuous random number generation test
    - DRBG continuous random number generation test
    - ECDSA pairwise consistency test (sign and verify)
    - RSA pairwise consistency test (sign and verify)
    - ECC CDH pairwise consistency test
    - Firmware Load Test: N/A
    - Bypass Test: N/A
    - Manual Key Entry Test: N/A

- **Upon successful completion of the power-up self-tests, the cryptographic module shall output a success status. Failure of any power-up self-tests results in the module entering an error condition (requires reset/re-login). The FIPS mode of operation indicator can be obtained through the Get Capabilities command and returns the following null-terminated status string: "FIPS-140"**

- **A user can initiate the power-up self-tests upon command after a RESET and restart of the cryptographic module. This process is part of the login process and cannot be skipped.**

- **There is no mechanism for firmware load. The firmware is fixed to the module either at chip manufacture or during pre-FIPS device provisioning.**

7. Identification and Authentication Policy

The module supports two roles: A Crypto Officer role and a User role.

The Crypto Officer role is limited to initialization functions that occur after its embedding in a device and during that device's manufacturing test and configuration:

- Booting the module for the first time
- If not already accomplished during module manufacturing (for example, during chip wafer test), seal the module firmware to the module
- Configuring the User Login verification CSPs: See Table 5 - Authentication elements
- Setting FIPS mode; this is a permanent change
- Locking out changes
- Rebooting the module in FIPS mode
- Causing the first User instance to be initialized

Module firmware changes are not possible in FIPS mode, and FIPS mode is irreversible.

The cryptographic module, once set to FIPS mode during integration and manufacture, supports a User role through identity-based authentication.

The Crypto Officer role takes the validation of an ECDSA SHA 256 signature over a random challenge from the ESM to complete its login.

The User role takes the validation of three pieces of data to complete a login:

- A 256-bit HMAC verification of a fixed data object
- A 256-bit HMAC verification of a variable data object consisting of a set of public keys
- An ECDSA-SHA-256 verification of a variable data object using one of the public keys from the previous step

Role	Type of authentication	Authentication data	Description
Crypto Officer	Identity-Based	256-bit ECDSA public key (transport public key)	The Crypto Officer in this case is the holder of the private key associated with the transport public key. The Crypto Officer seals login data to a module.
User	Identity-Based	256 Bit HMAC Key (Module Seal Key), 256 Bit HMAC (FW Key Set Seal Key), 256 Bit ECDSA Public Key (User Public Key)	The User, in this case. is the holder of the private key associated with the public key authentication data. The user loads keys and executes cryptographic algorithms.

Table 5 - Authentication elements

The strength of the implemented authentication mechanism exceeds the strength requirements imposed by FIPS 140-2. Table 6 - Authentication mechanisms declares the probabilities associated with random attempts and multiple consecutive attempts within a one-minute period to subvert the implemented authentication mechanism, per role.

Crypto Officer role

The system uses a one-phase authentication strategy (verification of an ECDSA SHA256 signature over a random challenge) for Crypto Officer login having a random probability p for success of $p = 1/(2^{256})$ or 8.64e-78.

Measurement shows that each attempt will take at least 12.6 ms (the time for signature validation), permitting no more than 4762 attempts per minute.

User role

The system uses a three phase authentication strategy for User login, each phase having a random probability p for success of $p = 1/(2^{256})$ or 8.64e-78 for each phase. The combined probability p_c is $p_c = p^3$ or 6.411e-232.

Measurement shows that each attempt takes at least 12.6ms, permitting no more than 4762 attempts per minute.

Authentication Strength

Role	Authentication mechanism	Strength of mechanism
Crypto Officer authentication	ECDSA P256 SHA-256 verification with transport public key	The probability that a random attempt will succeed reduces to finding a hash collision OR guessing the private key; both of which are $p = 1/(2^{256}) = 8.64e\text{-}78$. Therefore, the probability that a random attempt will succeed is 8.64e-78, which is less than 1 in 1,000,000. The probability that a random attempt in a one-minute period will succeed is 4762 x p = 4.1125e-74), which is less than 1 in 100,000.
User Authentication	HMAC verification with module seal key	The probability that a random attempt will succeed is $p = 1/(2^{256})$.
	HMAC verification with FW key set seal key	The probability that a random attempt will succeed is $p = 1/(2^{256})$.
	ECDSA P256 SHA-256 verification with User Public Key	The probability that a random attempt will succeed reduces to finding a hash collision OR guessing the private key; both of which are $p = 1/(2^{256})$.
	Aggregate	Totaling the three authentication mechanisms above for the user results in a combined probability p_c , where $p_c = p^3$ or 6.411e-232. Therefore, the probability that a random attempt will succeed is 6.411e-232, which is less than 1 in 1,000,000. The probability that a random attempt in a one-minute period will succeed is $1-(1-p^3)^{4762}$, or, using Taylor Approximation, 4762 x p^3= 3.0672e-228 , which is less than 1 in 100,000.

Table 6 - Authentication mechanisms

8. Service Access Control Policy

Definitions of types of access:

- I: Input, generally plain or cipher text or non-sensitive parameters
- IS: Input Sealed, Load a sealed private or secret key
- O: Output, generally plain or cipher text or non-sensitive results (for example, signatures)
- OS: Output Sealed, Wrap a loaded key for export to external storage.
- G: Generate (also new keys from a derive operation)
- Z: Zeroize
- U: Use
- Update: Update an internal state – e.g., for the DRBG

Sealed keys differ from wrapped keys in that they remain cryptographically bound to the device and cannot be used on any other device.

Role	Services	Cryptographic Keys and CSP's	Type of Access	Related Function _tag's
None	Startup[3]	Applet Public Key Applet Hash	U,I U,I	hc_Startup, hc_VerifyOTP
None	Get Capabilities (includes FIPS status)	None		hc_GetCapabilitiesBuffer
None	Seed/Reseed DRBG[4]	DRBG Seed DRBG Internal State	U U, Update	Internal only, no user callable function
None	Verify Sealed Login Data†	Module Seal Key	U,I	hc_VerifyBootloader
None	Verify Sealed Login Keyset†	Module Seal Key	U,I	hc_VerifyFwKeySet
None	Verify Signed Login Data	User Public Key	U,I	hc_VerifyFirmware
None	SHA	None	U,I,O	hc_HashInit, hc_HashAppend, hc_
None	ECDSA Verify	User Public Key	U,I	hc_EcDsaVerify
None	Get Challenge	DRBG Internal State	U, Update	hc_GetChallenge
None	Unlock	Transport Public Key	U,I	hc_Unlock

Table 7 - Unauthenticated services

†Services which may indirectly use the Module Key (for example, to regenerate the Instance Seal Key or the DIK Private Key).

[3] Self-Test occurs automatically at module start up and may not be bypassed.

[4] The seed/reseed function is called automatically during instantiation and as needed to comply with DRBG policies (e.g. when reseed_counter > reseed_interval).

Role	Services	Cryptographic Keys and CSP's	Type of Access	Related Function _tag's
Crypto Officer	Seal Login Data†	Module Seal Key	U,I,O	hc_SealBootloader
Crypto Officer	Seal Login Keyset†	Module Seal Key	U,I,O	hc_SealFwKeyset
Crypto Officer	Fuse	None		hc_Fuse
User, plus phys presence	Clear Instance	Instance DIK Private Key Instance Seal Key	Z Z	hc_Generate
User or Crypto Officer, plus phys presence	Initialize Instance	Instance DIK Private Key Instance Seal Key	G G	hc_Generate
User or Crypto Officer	Upgrade (changes which DIK is used)	Instance DIK Private Key Instance Seal Key	Z,G Z,G	hc_Upgrade
User, plus phys presence	Zeroize & Terminate Module†	All CSPs except Applet Hash (see Appendix)	Z	hc_Terminate
User or Crypto Officer	Get Device ID Public Key†	Instance DIK Private Key Instance DIK Public Key	U G,O	hc_GetDIKPubKey, hc_LoadDIK
User	Generate Key†	Ephemeral Shared Secret Instance Seal Key DRBG Internal State	G,OS U U, Update	hc_CreateRandomKey, hc_GenKwk
User	Generate KeyPair†	Ephemeral Private Key User Public Key Instance Seal Key DRBG Internal State	G, OS G, O U U, Update	hc_CreateRandomKey, hc_GetPubKey
User	Sign Data[5,6,†]	Ephemeral Private Key Instance DIK Private Key Ephemeral HMAC Key Instance Seal Key DRBG Internal State (for ECDSA only)	IS U IS U U, Update	hc_HashInit, hc_HashAppend, hc_HashFinal, hc_RsaSign, hc_EcDsaSign

[5] hc_Hash* functions include HMAC. HMAC verifications are done by regenerating the HMAC integrity tag (that is, by "Sign"ing) and comparing the result with the offered integrity tag.

[6] The ESM implements component Sign operations for RSA and ECDSA, which means that a hash over the data to be signed is calculated externally and then provided to the Sign function. The signature is output from the Sign function.

Role	Services	Cryptographic Keys and CSP's	Type of Access	Related Function _tag's
User	Verify Data[7]	User Public Key Ephemeral HMAC Key	I IS	hc_HashInit, hc_HashAppend, hc_HashFinal, hc_RsaVerify, hc_EcDsaVerify
User	Encrypt/Decrypt Data†	Ephemeral AES Key Instance Seal Key AES IV	IS,U U IS or I, U	hc_EncryptInit, hc_Encrypt, hc_EncryptFinal, hc_DecryptInit, hc_Decrypt, hc_DecryptFinal
User	Wrap/Unwrap Key†	Ephemeral AES Key Ephemeral HMAC Key Ephemeral Private Key Ephemeral Shared Secret	U,O,I O,I O,I O,I	hc_ImportKey
User	Derive Key†	User Public Key Ephemeral Private Key (of either EC or DH types) or Instance DIK Private Key Ephemeral Shared Secret Ephemeral AES Key AES IV Ephemeral HMAC Key Instance Seal Key	U,I U,OS G, OS, U G, OS OS G, OS U	hc_PerformKdf, hc_Ecdh, hc_Dh
User	Get Random	None	O	hc_GetRandom
User	Get Public Key†	Ephemeral Private Key User Public Key Instance Seal Key	IS O U	hc_GetPubKey
User	Save/Restore Context†	Context data Instance Seal Key	OS/IS U	hc_SaveContext, hc_RestoreContext

Table 8 - Authenticated services

[7] The ESM implements component Verify operations for RSA and ECDSA which means that a hash over the data to be verified is calculated externally and then provided to the Verify function along with the signature.
†Services which might indirectly use the Module Key (for example, to regenerate the Instance Seal Key or the DIK Private Key).

8.1. Description of Unauthenticated Services

The following services do not disclose, modify, or substitute any CSP, use an Approved security function, or otherwise affect the security of the cryptographic module except as an adjunct to the authentication process:

- Startup: The cryptographic module performs all of the required power-up self-tests.
- Get Capabilities: Returns the version information, FIPS mode, Self-test results and device state.
- Verify Sealed Login Data: Performs an HMAC verification of sealed login data to execute step one of the authentication process.
- Verify Sealed Login Keyset: Performs an HMAC verification of a sealed public key keyset to execute step two of the authentication process.
- Verify Signed Login Data: Performs an ECDSA-SHA-256 verification of signed data against one of the public keys retained from the Verify Sealed Login Keyset service to execute the third and final step of the authentication process.
- SHA: Performs a secure hash (using any of the supported and approved hash mechanisms) over a stream of data and returns the hash value.
- ECDSA Verify: Given a hash value, public key, and signature, returns a truth value indicating whether or not the signature verifies.
- Get Challenge: Obtain a fixed-length random nonce for use in the login process for the crypto officer.
- Unlock: Complete the Crypto Officer authentication process by providing control data and a signature over that control data and the previously returned challenge.

8.2. Description of Authenticated Services

The following services require either a Crypto Officer or User login and might have other prerequisites. Some commands are for use only during manufacturing:

- Seal Login Data: Create an integrity tag for the sealed login data.
- Seal Login Key Set: Create an integrity tag for a set of public keys for use in the User authentication process.
- Fuse: Finalize the manufacturing configuration of the SSN ESM.
- Clear Instance: Results in the zeroization of the instance seal key and the instance device identity keys. Has the side effect of rendering useless all keys sealed under that instance seal key. Module must be reset/rebooted after this action.
- Initialize Instance: Results in the creation of a new instance seal key and a set of instance device identity keys.
- Zeroize & Terminate Module: Zeroizes all CSPs in the module and optionally irreversibly renders the module inoperable for cryptographic operations, including those that do not involve sensitive operations such as signature verification and hash calculations.
- Get Device ID Public Key: Returns the public key associated with one of the Device Identity Key private keys.
- Generate Key: Generates a new AES key tagged for use only as a key wrapping key or an Ephemeral Shared Secret using the DRBG as a key stream source. The key is returned as a sealed object.
- Generate Key Pair: Generates a new EC or RSA key pair and returns the associated sealed private key. The public key can be derived or extracted from the sealed private key using the Get Public Key service.
- Sign Data: Signs the offered data or hash.
    - HMAC: Signs a stream of data and outputs an HMAC integrity tag of the appropriate length. An HMAC verify is done by repeating the sign operation and comparing the output with the original integrity tag.
    - ECDSA: Signs a hash value of a specific length for the defined EC curve and returns an ECDSA signature value.
    - RSA: Signs a pre-padded hash value of a specific length using RSA and returns an RSA signature.

- Verify Data: Verifies the offered data or hash against a signature or integrity tag.
    - ECDSA: Verifies an ECDSA signature over a provided hash value.
    - RSA: Verifies an RSA signature over a provided pre-padded hash value.
- Encrypt/Decrypt Data: Encrypts or decrypts a stream of data using a given symmetric key and mode and returns respectively either the ciphertext or plaintext.
- Wrap/Unwrap Key: Wraps or unwraps a private or secret key using a key wrapping key.

  The module and the User first perform key agreement using the Ephemeral Unified Model, C(2, 0, ECC CDH); User then imports the private or secret key (wrapped with "Ephemeral AES Key" tagged as a key wrapping key) into the module for use with crypto operations. The private or secret key is externalized and protected by the Instance Seal Key.
- Seal/Load Key: Wraps or unwraps a private or secret key using an internally generated, non-exportable symmetric key. The seal key is bound to the module and instance, and allows for the external storage and retrieval of material that is immutably bound to the seal key.
- Derive Key: Derives a private or secret key from a secret master key and some additional public data (Context and label). These functions are used to derive the pre-master secret (via asymmetric keys) and master secret (via KDF) to create session keys for endpoint communication.
- Get Random: Generates a stream of random octets from the internal DRBG.
- Get Public Key: Regenerates the matching public key from an existing sealed private key and returns the public key.
- Save/Restore Context
- Upgrade: Administratively marks one or more Device Identity keys as no longer being able to perform operations, while simultaneously marking another key as enabled to perform those operations.

9. Physical Security Policy

The physical security employed by the cryptographic module consists of embedding within a production grade IC packaging that is hard, opaque, and tamper evident. The following table describes the responsibilities and actions required by the operator to ensure that physical security is maintained.

Physical security mechanism	Recommended frequency of inspection/test	Inspection/test guidance details
Hard, opaque, tamper evident IC packaging	Upon receipt from chip manufacturer and again after circuit board integration and testing.	Perform a thorough visual inspection on all sides and angles of the enclosure for any signs of damage and malice including, but not limited to, scratches, scrapes, gouges, bent or broken pins, deterioration, and discoloring. If any suspicious characteristics are observed, return to the manufacturer or discard the cryptographic module.

Table 9 - Physical security mechanisms

NOTE: The module hardness testing was only performed at a single temperature and no assurance is provided for Level 3 hardness conformance at any other temperature.

10. Achieved Level of Security

The SSN ESM has achieved an overall security level of 3 for FIPS 140-2. The following table list the individual FIPS 140-2 areas and their corresponding level:

FIPS 140-2 Security Requirements Area	Level
Cryptographic Modules Specification	3
Cryptographic Module Ports and Interfaces	3
Roles, Services and Authentication	3
Finite State Model	3
Physical Security	3
Operational Environment	N/A
Cryptographic Key Management	3
EMI/EMC	3
Self-Tests	3
Design Assurance	3
Mitigation of Other Attacks	N/A

Table 10 - Module security levels

11. Mitigation of Other Attacks

The Mitigation of Other Attacks security section of FIPS 140-2 is not applicable to the Silver Spring Networks, Inc. Endpoint Security Module**.**

Other attacks	Mitigation mechanism	Specific limitations
N/A	N/A	N/A

Table 11 - Mitigation of other attacks

12. Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The SSN ESM hardware component cannot be certified by the FCC because it is not a standalone device. Rather, it is manufactured as a custom ASIC designed to be embedded into a set of electronic products which themselves would undergo standard EMI/EMC certification.

According to 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, the SSN ESM is not subject to EMI/EMC regulations because it is a subassembly designed for incorporation by equipment manufacturer into another device. That manufacturer is responsible for obtaining the necessary authorization for the equipment with the embedded SSN ESM prior to further marketing the product to a vendor or to a user.

13. References

[FIPS140-2] National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-2, May 25, 2001

[FIPS186-4] National Institute of Standards and Technology, *Digital Signature Standard (DSS)*, Federal Information Processing Standards Publication 186-4, July 2013

[FIPS197] National Institute of Standards and Technology, *Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication 197, November 26, 2001

[FIPS198-1] National Institute of Standards and Technology, *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standards Publication 198-1, July 2008

[SP800-38A] Dworkin, Morris; *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, NIST Special Publication 800-38A, December 2001

[SP800-38F] Dworkin, Morris; *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*, NIST Special Publication 800-38F

[SP800-56A] Barker, Elaine; Chen, Lily; et al.; *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, NIST Special Publication 800-56A Revision 2, May 2013

[SP800-90A] Barker, Elaine and Kelsey, John; *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, NIST Special Publication 800-90A Revision 1, June 2015

[SP800-108] Chen, Lily; *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*, NIST Special Publication 800-108, October 2009

[SP800-135] Dang, Quynh; *Recommendation for Existing Application-Specific Key Derivation Functions*, NIST Special Publication 800-135 Revision 1, December 2001

14. Appendix A: Critical Security Parameters and Public Keys

14.1. Transport Public Key

- Description: The public part of the Transport Key Pair. Used to verify a login (aka 'Unlock') by the Cryptographic Officer prior to FIPS for the purposes of sealing user login data to the module.
- Type: ECDSA (P-256)
- Generation: N/A - External HSM during Manufacturing
- Establishment: N/A
- Entry: N/A – Manual during Manufacturing Wafer Test
- Output: N/A
- Storage: Plaintext in OTP (Fuse) and HardCache Ram (HCRAM)
- Key-To-Key Entity: Crypto Officer
- Zeroization: N/A

14.2. Applet Public Key

- Description: The public part of the Applet Signing Key Pair. Used by the ESM bootloader to verify a signature over the applet directory prior to loading the applet code into the ESM.
- Type: ECDSA (P-256)
- Generation: N/A - External HSM during Manufacturing
- Establishment: N/A
- Entry: N/A – Manual during Manufacturing Wafer Test
- Output: N/A
- Storage: Plaintext in OTP (Fuse) and HardCache Ram (HCRAM)
- Key-To-Key Entity: Crypto Officer
- Zeroization: N/A

14.3. Applet Hash

- Description: Used to lock a specific applet directory to a module. This augments but does not replace the verification of the applet directory signature by the Applet Public Key. The module will fail to boot if there is a mismatch between the OTP Applet Hash and the ESM calculated hash over the to-be-loaded Applet Directory. While this item provides integrity protection over the operational environment of the module and is listed as a CSP for that reason, per FIPS140-2 IG, 7.17, because disclosure of this item (hash over public data) does not compromise the security of the module, and zeroization might, this item is not required to be zeroized on termination or otherwise.
- Type: Data (32 Bytes) (A SHA256 over the Applet Directory)
- Generation: External
- Establishment: N/A
- Entry: Manual–Wafer
- Output: No
- Storage: OTP (Fuse)
- Key-to-Key Entity: N/A
- Zeroization: N/A

14.4. Module Key

- Description: A 'master secret' for the module. This is generated once per module and stored in the SSN ESM OTP. Various symmetric and asymmetric internal keys are derived from this master key. This key was consciously excluded from many entries in the services tables above as it is only directly affected or used by the Zeroize service. Instead, keys derived from this key are regenerated as needed as an implicit side effect, and that regeneration might not necessarily be tied to a specific service. Those keys can be identified by a generation method of "KDF (module key)" in this section and a dagger (†) for those services that might involve the module key in the tables above.
- Type: Generic (64 bytes)
- Generation: SP 800-90A HMAC-SHA-512 DRBG; As per SP 800-133 Section 7.1, key generation is performed as per the "Direct Generation" of Symmetric Keys which is an Approved key generation method
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in OTP (Fuse) and HardCache Ram (HCRAM)
- Key-To-Key Entity: Process
- Zeroization: HCRAM Actively overwritten via the "Zeroize & Terminate Module" service; OTP set to all 1's

14.5. Module Seal Key

- Description: An HMAC key used to seal login data to the module.
- Type: HMAC-SHA-256
- Generation: KDF (Module Key); As per SP 800-133 Section 7.4, key derivation is performed using an Approved KDF (SP 800-108 KDF) with a key-derivation-key (Module Key) generated from an approved DRBG which is an Approved key generation method
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in HCRAM
- Key-To-Key Entity: Crypto Officer
- Zeroization: Actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.

14.6. FW Key Set Seal Key

- Description: An HMAC key used to seal login keyset data to the module.
- Type: HMAC-SHA-256
- Generation: KDF (Module Key); As per SP 800-133 Section 7.4, key derivation is performed using an Approved KDF (SP 800-108 KDF) with a key-derivation-key (Module Key) generated from an approved DRBG which is an Approved key generation method.
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in HCRAM
- Key-To-Key Entity: Crypto Officer
- Zeroization: Actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.

14.7. Instance Seal Key

- Description: An AES key wrap key used to protect user cryptographic material when it has been externalized from the module.
- Type: AES 256 Bit (SP 800-38F)
- Generation: KDF (Module Key); As per SP 800-133 Section 7.4, key derivation is performed using an Approved KDF (SP 800-108 KDF) with a key-derivation-key (Module Key) generated from an approved DRBG which is an Approved key generation method.
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in HCRAM
- Key-To-Key Entity: User
- Zeroization: Actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.

14.8. Instance Device Identity Key (DIK) Private Key

- Description: An asymmetric private key that is used to prove user/module identity for cryptographic protocols. The related public key is usually bound into a digital certificate.
- Type: ECDSA (P-256 or P-384)
- Generation: KDF (Module Key); Key derivation of value “K” is performed using an Approved KDF (SP 800-108 KDF) with a key-derivation-key (Module Key) generated from an approved DRBG which is an Approved key generation method. The ECDSA key pair is generated as per FIPS 186-4.
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in HCRAM
- Key-To-Key Entity: User
- Zeroization: Actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.

14.9. Instance Device Identity Key (DIK) Public Key

- Description: An asymmetric public key bound to the module and instance. The public part of the DIK key pair.
- Type: ECDSA (P-256 or P-384)
- Generation: Through calculation: X = xG
  where “X” is the public point of the public key, “G” is the generator point, and “x” is the private key scalar value from the Instance Device Identity Key Private Key.
- Establishment: N/A
- Entry: N/A
- Output: Yes–Clear
- Storage: Plaintext in HCRAM
- Key to Key Entity: User
- Zeroization: N/A

14.10. Context Data

- Description: Sealed (encrypted and integrity protected) intermediate state from a cryptographic operation that cannot be completed in a single command such as a hash or signature over large data.
- Type: Opaque Byte Data (variable)
- Generation: N/A
- Establishment: N/A
- Entry: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Output: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Storage: Plaintext in HCRAM; module will store this CSP in ARM RAM encrypted via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Key to Key Entity: User
- Zeroization: Plaintext value is actively overwritten during reset of the module, function completion or execution of the “Zeroize & Terminate Module” service.

14.11. DRBG Internal State

- Description: SP 800-90A HMAC-SHA-512 DRBG Internal State values “V” and “Key”
- Type: SP 800-90A HMAC-SHA-512 DRBG

  V – 512 bits

  Key – 512 bits
- Generation: SP 800-90A HMAC-SHA-512 DRBG
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in HCRAM
- Key-To-Key Entity: Process
- Zeroization: Plaintext value is actively overwritten during reset of the module, or execution of the “Zeroize & Terminate Module” service.

14.12. DRBG Seed

- Description: Seed material for SP 800-90A HMAC-SHA-512 DRBG consisting of entropy input provided by hardware TRNG and personalization string (fixed data stored inside the module boundary).
- Type: NDRNG (888 bits)
- Generation: SP 800-90A HMAC-SHA-512 DRBG
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Incorporated into DRBG state upon generation – plaintext is not recoverable
- Key-To-Key Entity: Process
- Zeroization: N/A

14.13. Ephemeral Private Key

- Description: Any user asymmetric private key of an approved type and key strength. Depending on the specific key type and generation parameters, the key may be used for key agreement or generation of digital signatures.

- Type: EC P-256 or P-384; RSA (2048 bits); DH (256 bits)

- Generation:

    - For RSA and EC Ephemeral Private Keys, the module will use SP 800-90A HMAC-SHA-512 DRBG to generate random value “K”; As per SP 800-133 Section 6.1, key generation is performed as per FIPS 186-4 which is an Approved key generation method.

    - For DH Ephemeral Private Keys, the module will use SP 800-90A HMAC-SHA-512 DRBG to generate random value “K”; As per SP 800-133 Section 6.2, key generation is performed as per SP 800-56A which is an Approved key generation method.

  In all of these cases, the generation of a private key also causes the generation or derivation of the related public key of the same type and strength using approved methods.

- Establishment: N/A

- Entry: Yes, encrypted and sealed via a valid key wrapping key using SP800-38F AES Key Wrapping mode

- Output: Yes, encrypted and sealed via the Instance Seal Key using SP800-38F AES Key Wrapping Mode as a side effect of the generation process. The public key is also output at this time in plain text.

- Storage: Plaintext in HCRAM or encrypted (SP800-38F AES KW) in external RAM or external non-volatile storage

- Key-To-Key Entity: User

- Zeroization: Plaintext value is actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.

14.14. User Public Key

- Description: Any user asymmetric public key of an approved type and key strength. Depending on the specific key type, the key may be used for key agreement or verification of digital signature. This class of key includes both the public keys generated in conjunction with an Ephemeral Private Key as well as public keys associated with private keys generated external to the module as they are indistinguishable for module purposes once the former is output from the module.
- Type: RSA (1024 bits- used for legacy signature verification only, 2048 bits); EC (P-256, P-384); DH (2048 Bits)
- Generation: External or in conjunction with the generation of an Ephemeral Private Key (see "Ephemeral Private Key" above)
- Establishment: N/A
- Entry: Yes, in plaintext via arguments to the Signature Verification and Key Agreement service calls
- Output: Yes, in plaintext in conjunction with the generation of an Ephemeral Private Key
- Storage: Plaintext in external RAM or external non-volatile storage
- Key-to-Key Entity: User
- Zeroization: N/A.

14.15. Ephemeral Shared Secret

- Description: The shared secret derived from the execution of the “Derive Key” service. Once generated, the secret is externalized and protected by the Instance Seal Key.
- Type: Generic master secret of 256, 384 or 2048 bits in length depending on the establishment method if derived using a key agreement mechanism (ECDH or DH), or any length appropriate if derived from another master secret.
- Generation: N/A
- Establishment:

  SP 800-56A Ephemeral Unified Model, C(2, 0, ECC CDH) or,

  SP 800-56A ECC CDH Primitive or,

  SP 800-56A FFC DH Primitive or,

  Any approved KDF
- Entry: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Output: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Storage: Plaintext in HCRAM
- Key-To-Key Entity: Process
- Zeroization: As per SP 800-56Ar1, this value is actively overwritten after the Derived Keying Material is generated by the module or actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.

14.16. Ephemeral HMAC Key

- Description: An HMAC key derived (using an approved KDF) from an ephemeral shared secret as needed for a cryptographic protocol. Each key generated, for all HMAC types, meet the FIPS 198-1 key length requirements of L/2. For HMAC-SHA-1, the minimum key size is 112 bits.
- Type: HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512
- Generation: N/A
- Establishment: Generated from an Ephemeral Shared Secret which is Approved as per SP 800-133 Section 7.3 via one of the following approved KDFs:

  SP 800-135 TLS V1.0 KDF or

  SP 800-135 TLS V1.2 KDF or

  SP 800-56A Concatenation KDF (Section 5.8.1) or

  ANSI X9.63 KDF or

  SP 800-108 Counter Mode KDF (HMAC-SHA-256) or

  SP 800-135 IKEv1 KDF (HMAC-SHA-256) or

  SP 800-135 IKEv2 KDF (HMAC-SHA-256)
- Entry: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Output: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Storage: Plaintext in HCRAM; module will store this CSP in Flash encrypted via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Key- To- Key Entity: User
- Zeroization: Plaintext value is actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.

14.17. Ephemeral AES Key

- Description: An AES key derived (using an approved KDF) from an ephemeral shared secret as needed for a cryptographic protocol or generated from the DRBG only for use as a key wrapping key.

- Type:

AES-128-ECB, AES-128-CBC, AES-128-CFB8, AES-128-OFB or

AES-192-ECB, AES-192-CBC, AES-192-CFB8, AES-192-OFB or

AES-256-ECB, AES-256-CBC, AES-256-CFB8, AES-256-OFB or AES-256 KW (SP 800-38F)

- Generation: DRBG (see below) – any key generated this way will be marked as a key wrapping key.

  SP 800-90A HMAC-SHA-512 DRBG; As per SP 800-133 Section 7.1, key generation is performed as per the "Direct Generation" of Symmetric Keys which is an Approved key generation method.

- Establishment: Generated from an Ephemeral Shared Secret which is Approved as per SP 800-133 Section 7.3 via one of the following approved KDFs:

  SP 800-135 TLS V1.0 KDF or

  SP 800-135 TLS V1.2 KDF or

  SP 800-56A Concatenation KDF (Section 5.8.1) or

  ANSI X9.63 KDF or

  SP 800-108 Counter Mode KDF (HMAC-SHA-256) or

  SP 800-135 IKEv1 KDF (HMAC-SHA-256) or

  SP 800-135 IKEv2 KDF (HMAC-SHA-256)

- Entry: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.

- Output: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.

- Storage: Plaintext in HCRAM; module will store this CSP in Flash encrypted via SP 800-38F AES Key Wrapping using the Instance Seal Key.

- Key- To- Key Entity: User

- Zeroization: Plaintext value is actively overwritten during reset of the module or execution of the "Zeroize & Terminate Module" service.

14.18. Ephemeral AES IV

- Description: An AES Initialization Vector derived (using an approved KDF) from an ephemeral shared secret as needed for a cryptographic protocol. AES modes supported by the module which can make use of this IV: CBC, CFB8, OFB.
- Type: Generic (128-bit)
- Generation: N/A
- Establishment: Generated from an Ephemeral Shared Secret which is Approved as per SP 800-133 Section 7.3 via one of the following approved KDFs:

  SP 800-135 TLS V1.0 KDF or

  SP 800-135 TLS V1.2 KDF or

  SP 800-56A Concatenation KDF (Section 5.8.1) or

  ANSI X9.63 KDF or

  SP 800-108 Counter Mode KDF (HMAC-SHA-256) or

  SP 800-135 IKEv1 KDF (HMAC-SHA-256) or

  SP 800-135 IKEv2 KDF (HMAC-SHA-256)
- Entry: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Output: Yes–Encrypted and sealed via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Storage: Plaintext in HCRAM; module will store this CSP in Flash encrypted via SP 800-38F AES Key Wrapping using the Instance Seal Key.
- Key- To- Key Entity: User
- Zeroization: Plaintext value is actively overwritten during reset of the module or execution of the “Zeroize & Terminate Module” service.
- AES mode: All AES modes supported (CBC,CFB8, OFB)

15. Appendix B: FIPS 140-2 IG, Section D.12 - Vendor Affirmation

In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) as per SP800-133 (vendor affirmed). The resulting generated seed, for asymmetric key generation, and the key for symmetric key algorithms are both from the unmodified output of an Approved SP 800-90A DRBG (Cert. #1906).