Self-Defending Key Management Service™

FIPS 140-2 Level 1 Non-Proprietary Security Policy

Date:	10/29/2018
Version:	2.0.596 and 2.0.NOAESNI-182
Document Number:	1.2

Table of Contents

Revision History

Author(s)	**Version**	**Date**	**Updates**
Fortanix, Inc.	**1.0**	**June 1, 2018**	**Initial Release**
Fortanix, Inc.	**1.1**	**October 3, 2018**	**Updates**
Fortanix, Inc.	**1.2**	**October 29, 2018**	**Updates**

1. Module Overview

Fortanix Self-Defending Key Management Service™ (SDKMS) is the world's first cloud service secured with Intel® SGX*. With SDKMS, you can securely generate, store, and use cryptographic keys and certificates, as well as secrets, such as passwords, API keys, tokens, or any blob of data. Your business-critical applications and containers can integrate with SDKMS using legacy cryptographic interfaces or using its native RESTful interface. SDKMS provides key management and cryptographic operations functionality via secure network interface. It provides access control to users and applications to enforce authorized access to keys.

FIPS 140-2 conformance testing was performed at Security Level 1. The following configuration was tested by the lab.

Platform	Module Name	Software Version	Operating System	Processor	Optimization
General purpose x86 based server (Supermicro SYS-5019S-MR)	Self-Defending Key Management Service™	2.0.596	Ubuntu 16.04	Intel® Xeon® CPU E3-1230 V5 @3.40GHz	AES-NI
General purpose x86 based server (Supermicro SYS-5019S-MR)	Self-Defending Key Management Service™	2.0.NOAESNI-182	Ubuntu 16.04	Intel® Xeon® CPU E3-1230 V5 @3.40GHz	None

Table 1 - Configurations tested

[*] NOTE: Intel® SGX includes cryptographic functionality that is not included within the logical cryptographic boundary, and as such all data items that are "obfuscated" by Intel® SGX are treated as plaintext from the perspective of the cryptographic module.

** NOTE: The CMVP allows porting of this cryptographic module from the operational environment specified on the validation certificate to an operational environment which was not included as part of the validation testing as long as the porting rules of FIPS 140-2 Implementation Guidance G.5 are followed. As per FIPS 140-2 Implementation Guidance G.5, no claim can be made as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed above in Table 1.

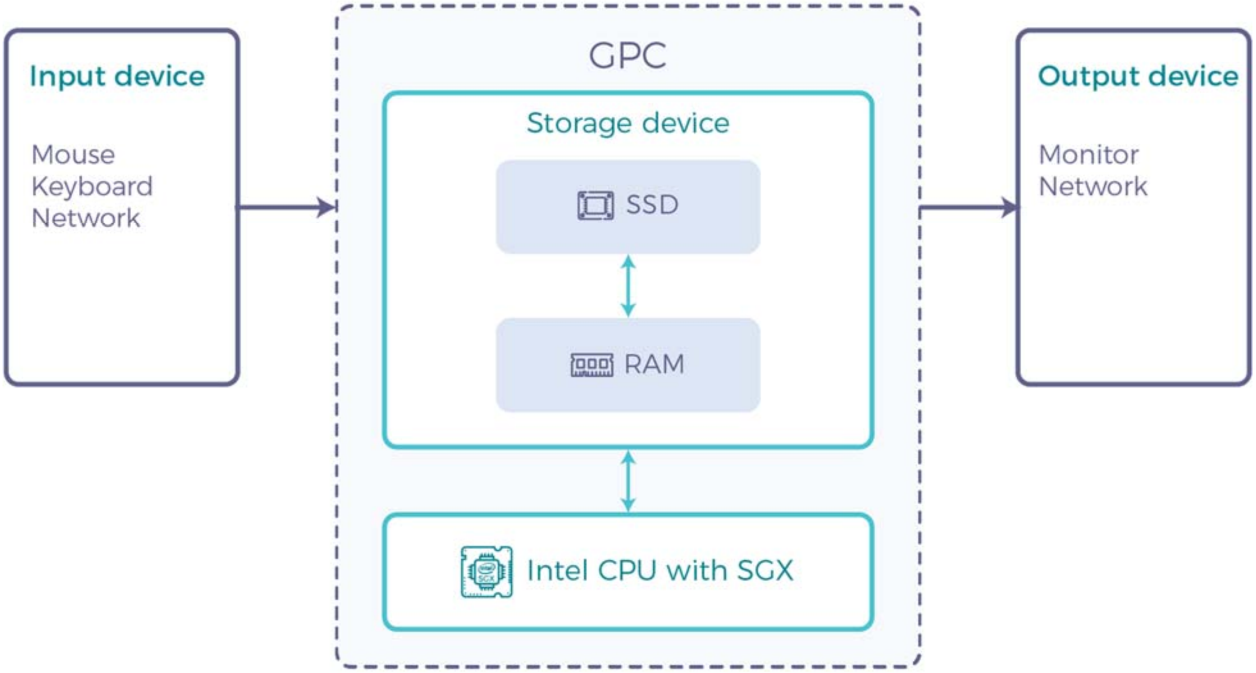
FIPS Security Area	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services and Authentication	3
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

Table 2- Security Level Specification Table

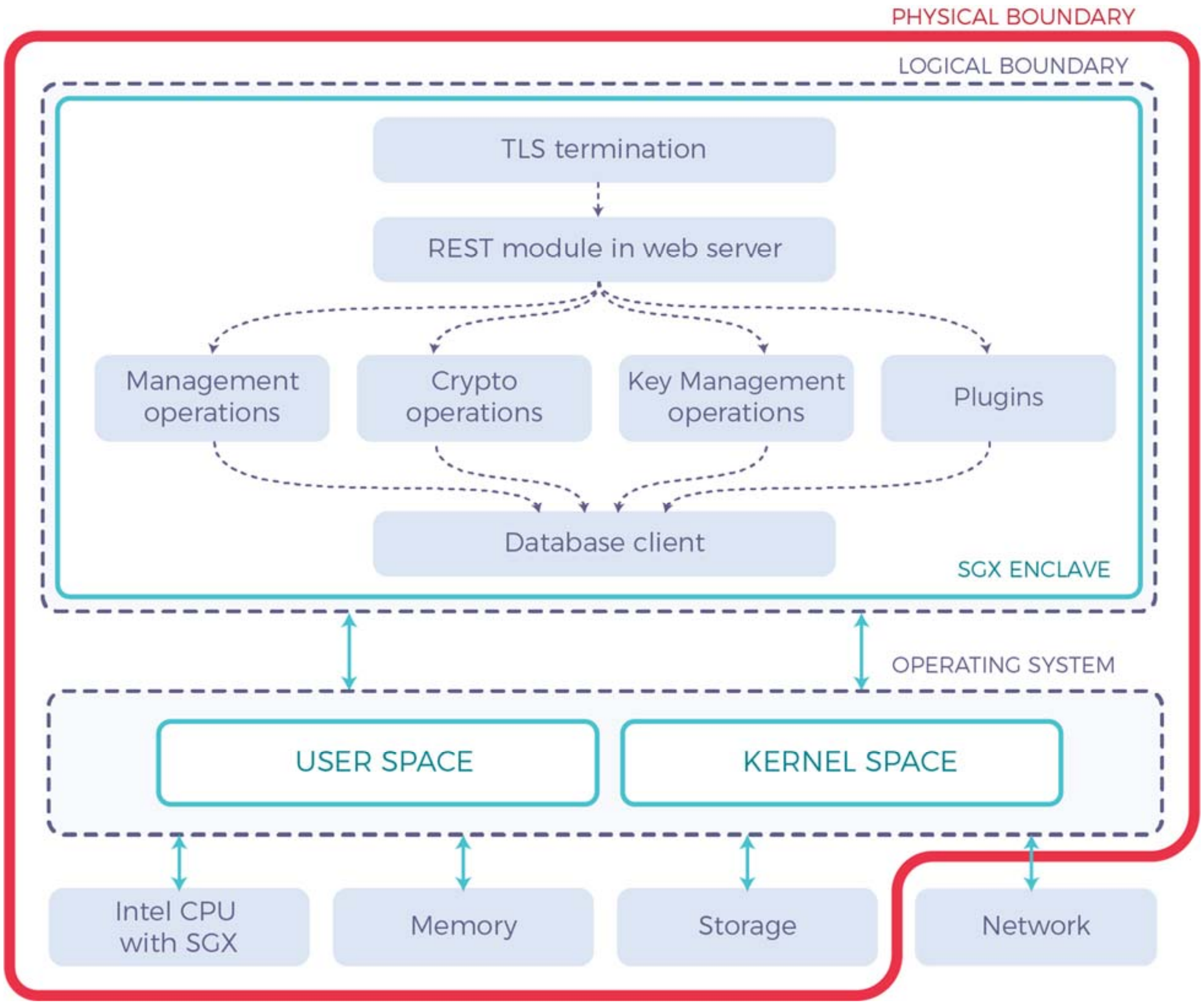
1.1 Cryptographic Boundary

The cryptographic module is a software-only module. The physical cryptographic boundary is the general-purpose computer on which the module is installed and runs. The physical embodiment of the module is a multiple-chip standalone cryptographic module. As a software module, the logical cryptographic boundary is the software module that compromises the various software components of the module.

1.1.1 Hardware Block Diagram



1.1.2 Software Block Diagram



2. Modes of Operations

The module supports two modes of operation: FIPS Approved mode and non-Approved mode.

2.1 FIPS Approved Mode

The Crypto Officer shall follow these steps to initialize the module and verify the module is running in the FIPS Approved Mode:

1. Power on the module
2. Install the module Debian package by running the following command
   a. For AES-NI version:
   ```
   sudo dpkg -i sdkms_2.0.596 -1_amd64.deb
   ```
   or for Non AES-NI version:
   ```
   sudo dpkg -i sdkms_2.0.NOAESNI-182 -1_amd64.deb
   ```
   b. After the package is installed, it will prompt you to complete the configuration. Using these prompts complete the configuration of the module.
3. Invoke the version API provided by the "Get status" service
4. Verify that the output is correct, with the following format and value of "fips_mode" attribute set to true:
   a. For AES-NI Version
   ```
   {"version":"2.0.596","api_version":"v1-20170718","
   server_mode":"Sgx","fips_mode":true}
   ```
   b. For Non AES-NI Version
   ```
   {"version":"2.0.NOAESNI-182","api_version":"v1-20170718","
   server_mode":"Sgx","fips_mode":true}
   ```

The module is now initialized and in the FIPS Approved Mode. Operators of the module must adhere to the Approved and Allowed Cryptographic Functions defined in this section, and to the Security Rules set forth in this Security Policy. Any deviation is an explicit violation of this Security Policy and implicitly toggles the module to the Non-Approved Mode regardless of the "fips_mode" attribute returned by the version API. Please see section Non-Approved Mode for more information.

2.1.1 Approved Cryptographic Functions

CAVP Cert #	Algorithm	Standard	Model/ Method	Key Lengths, Curves or Moduli	Use
5328, 5329, 5379, 5380	AES	FIPS 197, SP 800-38D, SP 800-38C	ECB, CBC, CTR, CFB 128, GCM, CCM	128, 192, 256	Data Encryption/ Decryption
1818, 1822	CVL	SP 800-135	KDF		Key Establishment
2072, 2073	DRBG	SP 800-90Ar1	CTR_DRBG With Derivation Function		Deterministic Random Bit Generation
1418, 1419	ECDSA	FIPS 186-4		P-192, P-224, P-256, P-384, P-521[1]	Key Pair Generation, Digital Signature Generation and Verification
3526, 3527	HMAC	FIPS 198-1	HMAC-SHA-1 HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	112, 128, 192, 256	Message Authentication
191, 195	KDF	SP 800-108	KDF		Key Derivation
5328 and 3526	KTS	SP 800-38F	AES CBC with HMAC-SHA-1	AES (128, 256) with	Key establishment methodology provides between 128 and 256

[1] In FIPS Approved mode, P-192 and SHA-1 are not allowed for ECDSA Signature Generation. The minimum hash sizes supported by the module are SHA-256 for P-224, SHA-256 for P-256, SHA-384 for P-384, and SHA-512 for P-521.

CAVP Cert #	Algorithm	Standard	Model/ Method	Key Lengths, Curves or Moduli	Use
			or HMAC-SHA-256	HMAC-SHA-1 (160) or HMAC-SHA-256 (256)	bits of encryption strength
5329 with 3527	KTS	SP 800-38F	AES CBC with HMAC-SHA-1 or HMAC-SHA-256	AES (128, 256) with HMAC-SHA-1 (160) or HMAC-SHA-256 (256)	Key establishment methodology provides between 128 and 256 bits of encryption strength
5379	KTS	SP 800-38F	GCM	128, 192, 256	Key establishment methodology provides between 128 and 256 bits of encryption strength
5380	KTS	SP 800-38F	GCM	128, 192, 256	Key establishment methodology provides between 128 and 256 bits of encryption strength

CAVP Cert #	Algorithm	Standard	Model/ Method	Key Lengths, Curves or Moduli	Use
2876, 2877	RSA	FIPS 186-4,	PKCS1 v1.5; GenKey9.31; PSS SHA-1, SHA-256, SHA-384, SHA-512	1024[2], 2048, 3072, 4096[3]	Digital Signature Generation and Verification
4280, 4281	SHS	FIPS 180-4	SHA-1, SHA-256, SHA-384 SHA-512		Message Digest

Table 3 - Table of Approved Algorithms

For additional information on transitions associated with the use of cryptography refer to NIST Special Publication SP 800-131Ar1. This document can be located on the CMVP website at: (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf).
The data in the tables will inform Users of the risks associated with using a particular algorithm and a given key length.

[2] In FIPS Approved mode, 1024-bit keys and SHA-1 are not allowed for RSA Signature Generation.

[3] As per FIPS 140-2 Implementation Guidance A.14 CAVP validation has been performed on key sizes testable via CAVS, while the cryptographic module supports any RSA modulus size between 2048 and 8192.

2.1.2 Not Approved but Allowed Algorithms

Algorithm	Caveat	Use
HMAC-MD5	Used as per SP 800-135 Rev1 Section 4.2.1	Only used in TLS V1.0/1.1 KDF
NDRNG	Only used to seed the CTR_DRBG With derivation function.	Seeding for the Approved DRBG
PBKDF	No Security Claimed	Used for obfuscation of passwords, considered as plaintext
RSA Key Wrapping	RSA (key wrapping; key establishment methodology provides 112 bits of encryption strength)	Key Wrapping

Table 4 - Table of Non-Approved but Allowed Algorithms

2.2 Non-Approved Mode

The module supports a Non-Approved Mode of operation. This mode of operation exists when the operator does not abide by the rules set forth in this Security Policy and invokes Non-Approved cryptographic algorithms or Non-Approved services described in this section. (Operator must abide by Security Rules in section Security Rules)

The use of any such algorithm, and service, is an explicit violation of this Security Policy and is explicitly disallowed by this Security Policy.

The algorithms marked "non-compliant" are not compliant because they are invoked in the Non-Approved mode of operation, by a Non-Approved mode service.

Non-Approved Algorithms	Usage/ Description
AES GCM (non-compliant)	• Encryption/Decryption with IV input from outside of the module
AES ECB (non-compliant), AES CBC (non-compliant), AES CTR (non-compliant), AES CFB 128 (non-compliant), AES CCM (non-compliant)	• Derive Key service using the resulting ciphertext as keying material
ECC CDH Primitive (non-compliant) CVL (Certs. #1840 and #1841)	• Agree Key service to calculate a shared secret
ECDSA (non-compliant)	• Sign/Verify using hash input created outside of the module • Sign using P-192 curve
RSA (non-compliant)	• 1024-bit RSA key SAML Authentication • Encryption/Decryption of data (SP 800-56B only allows for Key Encapsulation) • Sign/Verify using hash input created outside of the module • Sign/Verify using 1024-bit RSA key • Sign/Verify using 0x3 public key exponent

Table 5 Algorithms in Non-Approved Mode

Service	Module Role	Usage/Description	Algorithm Used
Agree Key	Application	• Calculate a shared secret	ECC CDH Primitive (non-compliant) CVL (Certs. #1840 and #1841)
Authentication	System Administrator, System Operator, Account Administrator, Account Member, Account Auditor, Group Administrator, Group Auditor	• 1024-bit RSA key SAML Authentication	RSA (non-compliant)
Derive key	Application	• Use the resulting ciphertext as keying material	AES ECB (non-compliant), AES CBC (non-compliant), AES CTR (non-compliant), AES CFB 128 (non-compliant), AES CCM (non-compliant)
Encrypt/ Decrypt	Application	• Encryption/Decryption with IV input from outside of the module	AES GCM (non-compliant)
	Application	• Encryption/Decryption of data (SP 800-56B only allows for Key Encapsulation)	RSA (non-compliant)
Sign/Verify	Application	• Sign/Verify using hash input created outside of the module • Sign/Verify using 1024-bit RSA key • Sign/Verify using 0x3 public key exponent	RSA (non-compliant)
	Application	• Sign/Verify using hash input created outside of the module • P-192 EC Curve	ECDSA (non-compliant)

Table 6 Services available in Non-Approved Mode

3. Ports and Interfaces

The module runs on a general-purpose computer with physical ports. The tested configurations include the following physical ports:

- 4 SATA3 (6Gbps) ports
- 2 RJ45 Gigabit Ethernet LAN ports
- 1 RJ45 Dedicated IPMI LAN port
- 4 USB 3.0 ports
- 2 USB 2.0 ports
- 1 VGA port
- 2 COM ports
- 2 SuperDOM (Disk on Module) ports with built-in power

The module does not include a maintenance interface.

The logical interface is various application programming interfaces (API). The logical interfaces of the module expose services that applications can call. The applications interacting with the module input control and data to the module through the input fields of the API and receive output data and/or status information via the output parameters of the API. API documentation describes in detail the successful operation output and error in case of a failed operation. Each of the FIPS 140-2 logical interfaces relates to the module's application programming interface as follows:

Logical Interface	Description
Data Input	Input / Request payload of API
Data Output	Output / Response payload of API
Control Input	API call
Status Output	API returning status information and return status codes provided by API

Table 7- Specification of Cryptographic Module Logical Interfaces

4. Roles, Services and Authentication

The module supports identity-based authentication for all roles. The module supports a Crypto Officer and User Role.

- The Crypto Officer installs and administers the module.
- The User uses the cryptographic services provided by the module. This role is assumed both by an actual user of the system and an external system that requires cryptographic services.

The module supports a variety of roles that are mapped to the two FIPS roles. Following table enumerates the mapping between module roles and FIPS roles:

Module Role	FIPS Role
System Administrator	Crypto Officer
System Operator	Crypto Officer
Account Administrator	Crypto Officer, User
Account Member	Crypto Officer, User
Account Auditor	Crypto Officer
Group Administrator	Crypto Officer, User
Group Auditor	Crypto Officer
Application	User
Node	Crypto Officer

Table 8 – Mapping of Module Roles to FIPS roles

4.1 Authenticated Services

The module provides the following services:

Service	Module Roles	Cryptographic Keys, CSPs and Public Keys	Types of Access to Cryptographic Keys and CSPs R – Read or Execute W – Write or Create Z – Zeroize
Authentication	**System Administrator** **System Operator** **Account Administrator** **Account Member** **Account Auditor** **Group Administrator** **Group Auditor** **Application** **Node**	User password	R
		API key	R
		RSA Public key of external Application	R
		User 2FA device public key	R
		Bearer token	R
		SAML – Idp public key	R
		Public key of an outside entity/server (Another SDKMS node)	R
Create/Generate key	**Account Administrator** **Group Administrator** **Application**	Database Wrapping key	W
		DRBG Entropy Input String	W
		DRBG Seed	R, W
		DRBG internal state	W
		Symmetric key	W
		HMAC key	W
		RSA private key for Digital Signatures	W
		RSA private key for Key Encapsulation Operations	W
		ECDSA private key	W
		ECDSA public key	W
		RSA public key for Key Encapsulation Operations	W
		RSA public key for Digital Signatures	W
Encrypt/Decrypt	**Application**	Symmetric key	R
		Cipher State Wrapping key	R, W
		SP 800-108 KDF internal state	R
		Account key	R
		Database Wrapping key	R
Sign/Verify	**Application**	Database Wrapping key	R
		RSA private key for Digital Signatures	R
		RSA public key for Digital Signatures	R

Service	Module Roles	Cryptographic Keys, CSPs and Public Keys	Types of Access to Cryptographic Keys and CSPs R – Read or Execute W – Write or Create Z – Zeroize
		ECDSA private key	R
		ECDSA public key	R
		ECDSA random number “k"	R, W
Wrap/Unwrap	**Application**	Database Wrapping key	R
		Symmetric key	R
		RSA private key for Key Encapsulation Operations	R
		RSA public key for Key Encapsulation Operations	R
HMAC	**Application**	Database Wrapping key	R
		HMAC key	R
Digest	**Application**	N/A	N/A
Import Key	**Account Administrator** **Group Administrator** **Application**	Database Wrapping key	R
		Symmetric key	W
		HMAC key	W
		RSA private key for Digital Signatures	W
		RSA private key for Key Encapsulation Operations	W
		RSA public key for Key Encapsulation Operations	W
		ECDSA private key	W
		ECDSA public key	W
		RSA public key for Digital Signatures	W
Export Key	**Account Administrator** **Group Administrator** **Application**	Database Wrapping key (This key is not exported)	R
		Symmetric key – if it was created or imported with export permission	R
		HMAC key – if it was created or imported with export permission	R
		RSA private key for Digital Signatures – if it was created or imported with export permission	R
		RSA public key for Digital Signatures – if it was created or imported with export permission	R

Service	Module Roles	Cryptographic Keys, CSPs and Public Keys	Types of Access to Cryptographic Keys and CSPs R – Read or Execute W – Write or Create Z – Zeroize
		RSA private key for Key Encapsulation Operations – if it was created or imported with export permission	R
		RSA public key for Key Encapsulation Operations – if it was created or imported with export permission	R
		ECDSA private key – if it was created or imported with export permission	R
		ECDSA public key – if it was created or imported with export permission	R
Cluster Management[4]	**System Administrator Node**	Cluster Master key	R, W
		Cluster RSA private key for TLS	R, W
		Cluster RSA public key for TLS	R, W
		Node RSA private key for SDKMS	R, W
		Node RSA public key for SDKMS	R, W
		Public key of an outside entity/server (CA)	R, W
		Public key of an outside entity/server (Another SDKMS node)	R, W
System configuration and management	**System Administrator System Operator (Read only) Account**	Cluster Master key	R
		System key	R, W
		Account Wrapping key	R, W
		Account key	R, W
		Database Wrapping key	R, W

[4] Please see “SDKMS Clusters” section for more information.

Service	Module Roles	Cryptographic Keys, CSPs and Public Keys	Types of Access to Cryptographic Keys and CSPs R – Read or Execute W – Write or Create Z – Zeroize
	Administrator Account Member Account Auditor (Read only) Group Administrator Group Auditor (Read only) Application	SP 800-108 KDF internal state	R, W
		User 2FA device public key	R, W
		SAML- Idp public key	R, W
		RSA Public key of external Application	R, W
TLS[5]	**System Administrator System Operator Account Administrator Account Member Account Auditor Group Administrator Group Auditor Application Node**	Cluster RSA private key for TLS	R
		Cluster RSA public key for TLS	R
		SP 800-135 TLS KDF internal state	R, W
		TLS integrity key (HMAC)	R,W
		TLS encryption key (AES)	R,W
		TLS pre-master secret	R,W
		TLS master secret	R,W
		Public key of an outside entity/server (CA)	R
		RSA Public key of external Application	R

Table 9 - Services Authorized for Roles, Access Rights within Services

[5] All API calls into the module are done over TLS V1.0/1.1 or TLS V1.2. No parts of these protocols, other than the KDFs, have been tested by the CAVP and CMVP.

4.2 Unauthenticated Services

Services
Get status
Run self-tests
Signup
Zeroization[6]

Table 10 - Unauthenticated Services

4.3 SDKMS Clusters

A Cluster is a group of SDKMS nodes. When a new SDKMS node is provisioned in the cluster, it will generate a key pair consisting of the Node RSA public key for SDKMS and Node RSA private key for SDKMS as well as a CSR. Once the node's certificate has been signed by the System Administrator using a CA, a credential for authenticating the new node in the cluster is created. The System Administrator then installs the signed node certificate and CA certificate in the node. Using its node certificate, the new node initiates a mutually authenticated TLS connection with an existing node in the cluster. Both nodes verify that the other party's node certificate is signed by the same CA, and the existing node sends the Cluster Master key over the TLS channel to the new node.

[6] The Crypto Officer of the module shall be physically present and in control of the module and the platform it is hosted in.

4.4 Authentication

The module supports the following authentication mechanisms.

Module Role	Authentication Type	Authentication Data
System Administrator System Operator Account Administrator Account Member Account Auditor Group Administrator Group Auditor	Identity Based	User password
Application	Identity Based	API key
Application	Identity Based	RSA Public key of external Application
System Administrator System Operator Account Administrator Account Member Account Auditor Group Administrator Group Auditor	Identity Based	User 2FA device public key
System Administrator System Operator Account Administrator Account Member Account Auditor Group Administrator Group Auditor Application	Identity Based	Bearer token

Module Role	Authentication Type	Authentication Data
System Administrator System Operator Account Administrator Account Member Account Auditor Group Administrator Group Auditor	Identity Based	SAML – Idp public key
Node	Identity Based	Public key of an outside entity/server (Another SDKMS node)

Table 11- Roles and required Identification and Authentication

Our password authentication policy is as described for the Memorized Secret Authenticators in NIST SP 800-63B (8 characters or longer). The module supports concurrent operators and the module levies a restriction on session expiry time where if inactive, the Application's role session will expire in 10 minutes by default. Similarly, for all other Module roles there is a session expiry time of 24 hours. Session expiry time can be customized.

Authentication Mechanism	Strength of Mechanism
User password	Minimum password length is 8 characters. For a user who just meets the minimum password length, each of the eight characters will have at least 95 possible characters[7] (ASCII printable characters with character code 32 -126) if we consider just the printable characters, although module supports UTF-8 characters for password and the number of possible characters with UTF-8 is much higher. Total number of password permutations with eight characters is 95^8 = 6,634,204,312,890,625. Therefore, the probability of guessing a password is significantly less than one in 1,000,000. Module only allows at the most 10 authentication attempts in a second. Therefore, a user could try at most 600 passwords in a minute. Given the total number of possible permutations (as shown above), the probability a random attempt in one-minute period to be correct will be 600/6,634,204,312,890,625. Therefore, the probability of guessing a password in a one-minute period is significantly less than one in 100,000.
SAML – Idp public key	The strength of this mechanism is based on the public key size that is used for signature verification. Minimum key size is RSA 2048, which provides at least security strength of 112 bits. Therefore, the

[7] Lower case and upper-case letters → 52 characters, Digits (0 to 9) → 10 characters, Special characters ~ ` ! @# $ % ^ & * ( ) _ - + = { } [ ] \ | ; : ' " < > , . ? / → 32 characters, Space → 1 character

Authentication Mechanism	Strength of Mechanism
	probability of success with random data is 1/(2^112), which is significantly less than one in 1,000,000. Module only allows at the most 10 authentication attempts in a second. Therefore, a user could try at most 600 attempts in a minute. Given the total number of possible permutations (as shown above), the probability a random attempt in one minute period to be correct will be 600/(2^112). Therefore, the probability of guessing a password in a one minute period is significantly less than one in 100,000.
API key	An application authenticates using an API key which contains app secret. App secret is a 64 bytes random data, which means 512 bits Therefore, total number of permutations for app secret will be 2^512. Therefore, the probability of guessing an application's secret is significantly less than one in 1,000,000. Module only allows at the most 10 authentication attempts in a second. Therefore, a user could try at most 600 attempts in a minute. Given the total number of possible permutations (as shown above), the probability a random attempt in one-minute period to be correct will be 600/(2^512). Therefore, the probability of guessing an app secrete in a one minute period is significantly less than one in 100,000.
User 2FA device public key	The module allows users to use a second factor authentication mechanism in addition to username and password. The strength of this combination mechanism

Authentication Mechanism	Strength of Mechanism
	relies upon the strength of the User password mechanism (described earlier) combined with the strength of two factor authentication. This mechanism adds more strength to the password mechanism which already far exceeds the FIPS requirements. U2F signature verification uses U2F device's public key which is an EC P-256 key. Security strength of this key is 128 bits. So the probability of a random success will be 1 in 2^128. Probability of this combined scheme = (Probability of guessing username and password) * (Probability from signature verification scheme), which is 1/(95^8) * 1/(2^128). Therefore, the probability of guessing a password is significantly less than one in 1,000,000. Module only allows at the most 10 authentication attempts in a second. Therefore, a user could try at most 600 attempts in a minute. Given the total number of possible permutations (as shown above), the probability a random attempt in one-minute period to be correct will be 600/(95^8 * 2^128). Therefore, the probability of guessing a password in a one-minute period is significantly less than one in 100,000. Therefore, this mechanism of additional 2FA also far exceeds the FIPS requirements.
RSA Public key of external Application	The strength of this mechanism is based on the size of the private key space. The module relies upon minimum RSA 2048-bit keys. This provides an encryption strength of 112 bits, so the probability of a random success will be 1 in 2^112, which is significantly less than one in 1,000,000. Using this mechanism, one can make very few attempts in one-minute period. Each attempt will require the module to check the signature on the certificate using FIPS approved

Authentication Mechanism	Strength of Mechanism
	signature algorithm and establishing TLS session with this certificate. On an average only one attempt can be made in a second. Therefore, at the most 60 attempts can be made in a one minute period. Therefore, the probability of guessing a 2048-bit private key and succeeding in a one minute period is 60/(2^112) which is significantly less than one in 100,000.
Bearer token	The bearer token is a base64 encoded random 64 bytes data which is generated using approved DRBG in SDKMS. This 64 bytes gives a total of 512 bits of data. Therefore, total number of permutations is 2^512. Therefore, the probability of guessing the token is 1/(2^512), which is significantly less than one in 1,000,000. Each authentication attempt takes approximately 12ms or more. Therefore, a user could try at most 5,000 attempts in a minute. Given the total number of possible permutations (as shown above), the probability a random attempt in one-minute period to be correct will be 5000/(2^512). Therefore, the probability of guessing a password in a one-minute period is significantly less than one in 100,000.
Public key of an outside entity/server (Another SDKMS node)	The strength of this mechanism is based on the size of the private key space. The module relies upon RSA 2048-bit node keys. This provides an encryption strength of 112 bits, so the probability of a random success will be 1 in 2^112, which is significantly less than one in 1,000,000. Each attempt will require the module to check the signature on the certificate using FIPS approved signature algorithm and establishing TLS session with this certificate. Each attempt takes 100ms or more. Therefore, at the most 600

Authentication Mechanism	Strength of Mechanism
	attempts can be made in a one minute period. Therefore, the probability of guessing a 2048-bit private key and succeeding in a one minute period is 600/(2^112) which is significantly less than one in 100,000.

Table 12 - Strength of Authentication Mechanisms

5. Self-tests

The module performs the following power-up and conditional self-tests. Upon successful execution of **all** power-up self-test, module provides the following status:

"Software Integrity test succeeded"
"Power-up self-tests succeeded"

Upon failure of a power-up or conditional self-test, the module halts its operation and enters the error state. The following tables describe self-tests implemented by the module along with status messages.

5.1 Power-Up Self Tests

Algorithm	Test	Status
AES 128-bit key size in ECB, CBC, CFB128, and CTR Modes 192-bit key size ECB, CBC, and CFB128 Modes 256-bit key size ECB, CBC, and CFB128 Modes	KAT (encryption)	Success: "*Power-up self-tests succeeded*" Error: "*AES self test failed*"
AES 128-bit key size in ECB, CBC, CFB128, and CTR Modes	KAT (decryption)	Success: "*Power-up self-tests succeeded*" Error: "*AES self test failed*"

Algorithm	Test	Status
192-bit key size ECB, CBC, and CFB128 Modes 256-bit key size ECB, CBC, and CFB128 Modes		
AES GCM 128-bit, 192-bit, and 256-bit key size	KAT (encryption)	Success: "*Power-up self-tests succeeded*" Error: "*GCM self test failed*"
AES GCM 128-bit, 192-bit, and 256-bit key size	KAT (decryption)	Success: "*Power-up self-tests succeeded*" Error: "*GCM self test failed*"
AES CCM 128-bit key size	KAT (encryption)	Success: "*Power-up self-tests succeeded*" Error: "*CCM self test failed*"
AES CCM 128-bit key size	KAT (decryption)	Success: "*Power-up self-tests succeeded*" Error: "*CCM self test failed*"
ECC CDH Primitive "Z" P-224 Curve	KAT	Success: "*Power-up self-tests succeeded*" Error: "*KAS ECC Primitive Z test failed*"
SHA-1	KAT	Success: "*Power-up self-tests succeeded*" Error: "*SHA1 self test failed*"
SHA-256	KAT	Success: "*Power-up self-tests succeeded*" Error: "*SHA256 self test failed*"
SHA-512	KAT	Success: "*Power-up self-tests succeeded*" Error: "*SHA512 self test failed*"

Algorithm	Test	Status
HMAC-SHA-1 128-bit key size	KAT	Success: "*Power-up self-tests succeeded*" Error: *"HMAC SHA1 self test failed"*
HMAC-SHA-256 128-bit key size	KAT	Success: "*Power-up self-tests succeeded*" Error: *"HMAC SHA256 self test failed"*
HMAC-SHA-512 2048-bit key size	KAT	Success: "*Power-up self-tests succeeded*" Error: *"HMAC SHA512 self test failed"*
SP 800-90A DRBG	KAT	Success: "*Power-up self-tests succeeded*" Error: *"CTR DRBG self test failed"*
RSA 2048-bit key size, SHA-256 (PKCS1 v1.5)	Signature generation/verification KAT	Success: "*Power-up self-tests succeeded*" Error: *"RSA self test failed"*
ECDSA P-224 curve	Signature generation/verification pairwise consistency test	Success: "*Power-up self-tests succeeded*" Error: *"ECDSA self test failed"*
SP 800-135 TLS V1.0/1.1 KDF	KAT	Success: "*Power-up self-tests succeeded*" Error: *"TLS 1.0 KDF self test failed"*
SP 800-135 TLS V1.2 KDF	KAT	Success: "*Power-up self-tests succeeded*" Error: *"TLS 1.2 KDF self test failed"*
SP 800-108 KDF 256-bit key size	KAT	Success: "*Power-up self-tests succeeded*" Error: *"KDF108 self test failed"*
HMAC-SHA-256 256-bit key size	Software integrity test	Success: *"Software Integrity test succeeded"* Error: *"Software integrity check failed"*
Critical Functions Tests	N/A	N/A

Table 13 – Power-Up Self-tests

5.2 Conditional Self Tests

Algorithm	Test	Status
Continuous RNG test performed on output of NDRNG (RDSEED)	Continuous Random Number Generator (RNG) Test	Error: *"FIPS conditional test failure: Error in cryptographic operation – RNG failed"*
Continuous RNG test performed on output of software-based Approved SP 800-90A CTR_DRBG	Continuous Random Number Generator (RNG) Test	Error: *"FIPS conditional test failure: Error in cryptographic operation – RNG failed"*
RSA 2048-bit to 8192-bit key size SHA-256	Pairwise Consistency Test (Sign and Verify)	Error: *"FIPS conditional test failure: Pairwise consistency test failed. Sign / Verify test failed."*
RSA 2048-bit to 8192-bit key size	Pairwise Consistency Test (Encrypt and Decrypt)	Error: *"FIPS conditional test failure: Pairwise consistency test failed. Encryption / Decryption test failed."*
ECDSA P-224, P-256, P-384, P-521 SHA-256	Pairwise Consistency Test (Sign and Verify)	Error: *"FIPS conditional test failure: Pairwise consistency test failed. Sign / Verify test failed."*
Bypass Test	N/A	N/A
Software Load Test	N/A	N/A
Manual Key Entry Test	N/A	N/A

Table 14- Conditional Self-tests

6. Physical Security

The module is a software-only module, so the physical security requirements of FIPS 140-2 Area 5 do not apply.

Physical Security Mechanisms	Recommended Frequency of Inspection/Test	Inspection/Test Guidance Details
N/A	N/A	N/A

Table 15- Inspection/Testing of Physical Security Mechanisms

7. Mitigation of Other Attacks Policy

The cryptographic module is not designed to mitigate any other attacks beyond the specific scope of FIPS 140-2.

Other Attacks	Mitigation Mechanism	Specific Limitations
N/A	N/A	N/A

Table 16- Table of Mitigation of Other Attacks

8. Security Rules

1. The module enforces logical separation between all data inputs, data outputs, control inputs, and status outputs via the cryptographic module API.
2. The cryptographic module inhibits all data output during self-tests and error states. The data output interface is logically disconnected from the processes performing self-tests and zeroization.
3. The cryptographic module runs on a general-purpose computing platform that conforms to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B (i.e. for Home use) which vacuously satisfies Class A.
4. Power-up self-tests do not require any operator intervention (i.e. the cryptographic module includes a default entry point as per FIPS 140-2 Implementation Guidance 9.10).
5. Power-up self-tests may be initiated on demand by power-cycling the module.
6. The cryptographic module does not support a maintenance interface or maintenance role.
7. The cryptographic module does not support manual key entry.
8. The cryptographic module does not support a bypass capability.
9. The cryptographic module does not support a Software Load Test.
10. The general-purpose computing platform includes a power port.
11. The cryptographic module supports both a FIPS-Approved mode of operation and Non-Approved mode of operation.
12. Results of previous authentications are cleared when the module is powered off. The operator is required to re-authenticate into the module.
13. The operator can Power cycle the module in order to exit the error states and resume normal operation. Otherwise, reinstall the module onto the general-purpose computing platform.
14. The module protects public keys and CSPs from unauthorized disclosure, unauthorized modification, and unauthorized substitution.
15. The module does not output intermediate key values.
16. When performing zeroization, the Crypto Officer of the module shall be physically present and in control of the module and the platform it is hosted in. The Crypto Officer is **required** to reformat and overwrite the platform's hard drive completely and **must** reboot the platform upon completion.
17. The module is an application implemented in client/server architecture, whereby the module is implemented in a server environment. Therefore, as per FIPS 140-2 IG Section 6.1, the server application is the single-user of the cryptographic module.
18. It is the authorized operator's responsibility to ensure that a key is used for one given purpose.
19. As per SP 800-56B, RSA encryption shall only be used for key wrapping.
20. The module complies with FIPS 140-2 IG A.5 requirements for AES-GCM:
    a. For TLS V1.2 Protocol, the module constructs the IV (internally) as allowed per Technique #1 in FIPS 140-2 IG A.5 for Industry Protocols. The IV total length is 96-bits, where the fixed IV length is 32-bits and nonce_explicit part of the IV is

64-bits. The GCM key and IV are session specific; if the module loses power the implementation is required to re-initialize a TLS V1.2 session, creating a new IV altogether.

   b. For the Encrypt/Decrypt service, a 96-bit IV is constructed from the output of the CTR_DRBG, allowed as per Technique #2 in FIPS 140-2 IG A.5 for IVs generated “internally at its entirety randomly”. In case the module’s power is lost and then restored, a new key for use with the AES GCM encryption/decryption will be generated from the output of the CTR_DRBG.

21. The operator is prohibited from entering AES-GCM IVs in FIPS-Approved mode.
22. Requests to use the 0x3 public key exponent for RSA are not allowed. The operator shall only use the 0x10001 public key exponent which is offered by default in the FIPS Approved Mode.
23. The operator shall use Tag Lengths greater than or equal to 64-bits for AES-GCM and AES-CCM.
24. In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) as per SP 800-133 (Vendor Affirmed). The resulting generated symmetric key and/or generated seed for asymmetric key generation, are from the unmodified output of the SP 800-90A DRBG.
25. The operator of the module shall abide by the requirements of FIPS 198-1 and SP 800-57 when executing the HMAC service:
    a. 112-bit HMAC key minimum for HMAC-SHA-1
    b. 128-bit HMAC key minimum for HMAC-SHA-256
    c. 192-bit HMAC key minimum for HMAC-SHA-384
    d. 256-bit HMAC key minimum for HMAC-SHA-512
26. Each call to the entropy source (RDSEED), which is within the physical boundary and outside the logical boundary, provides 8 bytes (64 bits) of entropy. Therefore, the minimum bits of entropy requested per each GET function call is 64 bits.
27. Module WebUI displays a dot (.) for each character of password entered to obscure feedback of the authentication data to an operator during entry of the authentication data.

9. Appendix A: CSPs

1. Cluster Master key
   - Description: 256-bit Key Derivation key (SP 800-108 KDF) used to derive the System key and Account Wrapping key
   - Generation: SP 800-90A CTR_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: Automatic, encrypted over TLS session (with TLS encryption key (AES)) for the “Cluster Management” service
   - Storage: Plaintext in RAM, plaintext in persistent storage
   - Key-to-Entity: This key belongs to the cluster
   - Zeroization: Procedural

2. System key
   - Description: 256-bit AES GCM key used to wrap all user and session information that is stored in persistent storage
   - Generation: Derived from Cluster Master key using NIST SP 800-108 KDF in Feedback Mode (§5.2); As per SP 800-133 Section 7.4, key derivation is performed by an Approved KDF which is an Approved key derivation method
   - Establishment: N/A
   - Entry: N/A
   - Output: N/A
   - Storage: Plaintext in RAM
   - Key-to-Entity: This key belongs to the cluster
   - Zeroization: Procedural

3. Account Wrapping key
   - Description: 256-bit AES GCM key used to wrap Account key when it is stored in persistent storage
   - Generation: Derived from Cluster Master key using NIST SP 800-108 KDF in Feedback Mode (§5.2); As per SP 800-133 Section 7.4, key derivation is performed by an Approved KDF which is an Approved key derivation method
   - Establishment: N/A
   - Entry: N/A
   - Output: N/A

- Storage: Plaintext in RAM
- Key-to-Entity: This key belongs to the cluster
- Zeroization: Procedural

4. Account key
   - Description: 256-bit Key Derivation key (SP 800-108 KDF) used to derive the Database Wrapping key and Cipher State Wrapping key
   - Generation: SP 800-90A CTR_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 RAM, encrypted in persistent storage with AES-GCM-256 Account Wrapping key
   - Key-to-Entity: This key belongs to a specific account / tenant and is unique to every account
   - Zeroization: Procedural

5. Database Wrapping key
   - Description: 256-bit AES GCM key used to wrap all account / tenant data and keys that belong to a specific account / tenant when it is stored in persistent storage
   - Generation: Derived from Account key using NIST SP 800-108 KDF in Feedback Mode (§5.2); As per SP 800-133 Section 7.4, key derivation is performed by an Approved KDF which is an Approved key derivation method
   - Establishment: N/A
   - Entry: N/A
   - Output: N/A
   - Storage: Plaintext in RAM
   - Key-to-Entity: This key belongs to a specific account / tenant and is unique to every account
   - Zeroization: Procedural

6. Cipher State Wrapping key
   - Description: 128-bit AES GCM key used to wrap all cipher state data that belongs to a specific account / tenant

- Generation: Derived from Account key using NIST SP 800-108 KDF in Feedback Mode (§5.2); As per SP 800-133 Section 7.4, key derivation is performed by an Approved KDF which is an Approved key derivation method
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: This key belongs to a specific account / tenant and is unique to every account
- Zeroization: Procedural

7. Symmetric key

- Description: 128-bit, 192-bit, or 256-bit AES keys in the following modes:
  - ECB
  - CBC
  - CTR
  - CFB 128
  - GCM
  - CCM Mode
- Generation: SP 800-90A CTR_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: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
- Zeroization: Procedural

8. HMAC key

- Description: HMAC key with the following key sizes:
  - For HMAC-SHA-1, the minimum key size is 112-bits.
  - For HMAC-SHA-256, the minimum key size is 128-bits.
  - For HMAC-SHA-384, the minimum key size is 192-bits.

  - For HMAC-SHA-512, the minimum key size is 256-bits.
- Generation: SP 800-90A CTR_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: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
- Zeroization: Procedural

9. RSA private key for Digital Signatures

- Description: 2048-bit to 8192-bit RSA key
- Generation: SP 800-90A CTR_DRBG; this key is used for Digital Signature Generation. As per SP 800-133 Section 6.1, key generation is performed as per FIPS 186-4 which is an Approved key generation method
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
- Zeroization: Procedural

10. RSA private key for Key Encapsulation Operations

- Description: 2048 to 8192-bit RSA key
- Generation: SP 800-90A CTR_DRBG; this key is used for Key Un-encapsulation (decryption) operations. This is an allowed method for key transport as per FIPS 140-2 IG D.9
- Establishment: N/A

- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
- Output: Automatic, Encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
- Zeroization: Procedural

11. ECDSA private key

- Description: EC Key (P-224, P-256, P-384, P-521)
- Generation: SP 800-90A CTR_DRBG; As per SP 800-133 Section 6.1, key generation is performed as per FIPS 186-4 which is an Approved key generation method
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
- Zeroization: Procedural

12. ECDSA random number “k”

- Description: A secret random number generated via SP 800-90A CTR_DRBG for use during the ECDSA signature generation process. The sizes are as follows:
  - For P-224, k is 224 bits.
  - For P-256, k is 256 bits.
  - For P-384, k is 384 bits.
  - For P-521, k is 521 bits.
- Generation: SP 800-90A CTR_DRBG; As per SP 800-133 Section 6.1, key generation is performed as per FIPS 186-4 which is an Approved key generation method
- Establishment: N/A
- Entry: N/A
- Output: N/A

- Storage: Plaintext in RAM
- Key-to-Entity: Process - “Sign/Verify” service with ECDSA
- Zeroization: Procedural

13. Cluster RSA private key for TLS

- Description: 2048-bit RSA key; when the module behaves as a TLS Server this key is used for RSA Key Un-encapsulation of the TLS pre-master secret
- Generation: SP 800-90A CTR_DRBG; As per SP 800-133 Section 6.2, key generation is performed as per FIPS 186-4; this is an allowed method as per FIPS 140-2 IG D.9
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 System key
- Key-to-Entity: This key belongs to the cluster
- Zeroization: Procedural

14. SP 800-135 TLS KDF internal state

- Description: 128-byte internal state for SP 800-135 TLS V1.0/1.1 KDF (HMAC-MD5/HMAC-SHA-1 PRF) or TLS V1.2 KDF (HMAC-SHA-256 PRF or HMAC-SHA-384 PRF)
- Generation: N/A
- Establishment: SP 800-135 Section 4.2.1 or 4.2.2; allowed method as per FIPS 140-2 IG D.8 Scenario 4
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: Process – TLS KDF internal state
- Zeroization: Procedural

15. TLS integrity key (HMAC)

- Description: 160-bit HMAC-SHA-1 key or 256-bit HMAC-SHA-256 key
- Generation: Derived from TLS master secret using SP 800-135 KDF Section 4.2.1 or 4.2.2; allowed method as per FIPS 140-2 IG D.8 Scenario 4
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM

- Key-to-Entity: Process – TLS
- Zeroization: Procedural

16. TLS encryption key (AES)

- Description: AES with the following modes and key sizes:
    - AES-128-CBC
    - AES-128-GCM
    - AES-128-CCM
    - AES-128-CCM with 64-bit Tag Length
    - AES-256-CBC
    - AES-256-GCM
    - AES-256-CCM
    - AES-256-CCM with 64-bit Tag Length
- Generation: Derived from TLS master secret using SP 800-135 KDF Section 4.2.1 or 4.2.2; allowed method as per FIPS 140-2 IG D.8 Scenario 4
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: Process – TLS
- Zeroization: Procedural

17. TLS pre-master secret

- Description: 48-byte pre-master secret
- Generation: SP 800-90A CTR_DRBG; generated only when the module behaves as a TLS Client. 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: When the module behaves as a TLS Server, the module may receive this secret RSA Key Encapsulated with "Cluster RSA public key for TLS". This is allowed as per FIPS 140-2 IG D.9
- Output: When the module behaves as a TLS Client, the module may output this value RSA Key Encapsulated with "Public key of an outside entity / server (Another SDKMS node)". This is allowed as per FIPS 140-2 IG D.9
- Storage: Plaintext in RAM
- Key-to-Entity: Process – TLS
- Zeroization: Procedural

18. TLS master secret

- Description: 48-byte master secret
- Generation: Derived from TLS pre-master secret using SP 800-135 KDF Section 4.2.1 or 4.2.2; allowed method as per FIPS 140-2 IG D.8 Scenario 4
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: Process – TLS
- Zeroization: Procedural

19. DRBG Entropy Input String

- Description: 384-bit Entropy Input String output from NDRNG (RDSEED)[8]
- Generation: Internally generated by the NDRNG (RDSEED)
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: Process - DRBG
- Zeroization: Procedural

20. DRBG Seed

- Description: 384-bit DRBG Entropy Input String XOR with personalization string and processed by derivation function
- Generation: SP 800-90A CTR_DRBG (AES-256) with Derivation Function
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: Process - DRBG
- Zeroization: Procedural

[8] The software module contains an approved CTR_DRBG that is seeded exclusively from one known entropy source (RDSEED) located within the operational environment inside the module’s physical boundary but outside the logical boundary.

21. DRBG internal state

- Description: Value of V (128-bits) and Key (256-bits) for SP 800-90A CTR_DRBG (AES-256) with Derivation Function
- Generation: SP 800-90A CTR_DRBG (AES-256) with Derivation Function
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: Process - DRBG
- Zeroization: Procedural

22. SP 800-108 KDF internal state

- Description: 256-bit internal state for SP 800-108 KDF in Feedback Mode (§5.2) with HMAC-SHA-256
- Generation: SP 800-108 KDF in Feedback Mode (§5.2); As per SP 800-133 Section 7.4, key derivation is performed by an Approved KDF which is an Approved key derivation method
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: Internal state
- Zeroization: Procedural

23. Node RSA private key for SDKMS

- Description: 2048-bit RSA key to support node-to-node communication with mutual authentication
- Generation: SP 800-90A CTR_DRBG; This key is used for RSA Key Un-encapsulation of the TLS pre-master secret. As per SP 800-133 Section 6.2, key generation is performed as per FIPS 186-4; this is an allowed method as per FIPS 140-2 IG D.9
- Establishment: N/A
- Entry: N/A
- Output: N/A
- Storage: Plaintext in RAM, plaintext in persistent storage
- Key-to-Entity: This key belongs to a specific node
- Zeroization: Procedural

24. User password

- Description: String of ASCII characters with a minimum of 8 bytes
- Generation: N/A - Entered by user
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Authentication" service
- Output: N/A
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 System key
- Key-to-Entity: This CSP belongs to a specific user; PBKDF2 resulting key is stored along with user object for future authentication
- Zeroization: Procedural

25. API key

- Description: 64-byte application authentication data
- Generation: SP 800-90A CTR_DRBG
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Authentication" service
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) for "System configuration and management" service
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: This belongs to a specific application
- Zeroization: Procedural

26. Bearer token

- Description: 64-byte authentication data
- Generation: SP 800-90A CTR_DRBG
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Authentication" service and invocation of all subsequent authenticated services thereof
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) for "Authentication" service
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 System key
- Key-to-Entity: This belongs to a specific authenticated session

- Zeroization: Procedural

10. Appendix B: Public Keys

1. RSA public key for Digital Signatures

- Description: 2048-bit to 8192-bit RSA key
- Generation: SP 800-90A CTR_DRBG; this key is used for Digital Signature Verification. As per SP 800-133 Section 6.1, key generation is performed as per FIPS 186-4 which is an Approved key generation method
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
- Zeroization: N/A

2. RSA public key for Key Encapsulation Operations

- Description: 2048-bit to 8192-bit RSA key
- Generation: SP 800-90A CTR_DRBG; this key is used for Key Encapsulation operations. This is an allowed method for key transport as per FIPS 140-2 IG D.9
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
- Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
- Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
- Zeroization: N/A

3. ECDSA public key
   - Description: EC key (P-224, P-256, P-384, P-521)
   - Generation: SP 800-90A CTR_DRBG; As per SP 800-133 Section 6.1, key generation is performed as per FIPS 186-4 which is an Approved key generation method
   - Establishment: N/A
   - Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Import Key" service
   - Output: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "Export Key" service if the key was created or imported with export permission
   - Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 Database Wrapping key
   - Key-to-Entity: Only authenticated clients can request the use of the key and authorization to access the key is checked. Client making the request must have authorization to use the key
   - Zeroization: N/A

4. Cluster RSA public key for TLS
   - Description: 2048-bit RSA key
   - Generation: SP 800-90A CTR_DRBG; when the module is a TLS Server, this key is used for RSA Key Encapsulation of the TLS pre-master secret. As per SP 800-133 Section 6.2, key generation is performed as per FIPS 186-4; this is an allowed method as per FIPS 140-2 IG D.9
   - Establishment: N/A
   - Entry: N/A
   - Output: Plaintext during TLS handshake
   - Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 System key
   - Key-to-Entity: This key belongs to the cluster
   - Zeroization: N/A

5. Node RSA public key for SDKMS
   - Description: 2048-bit RSA key to support node-to-node communication with mutual authentication
   - Generation: SP 800-90A CTR_DRBG; when the module is a TLS Server, this key is used for RSA Key Encapsulation of the TLS pre-master secret. As per SP 800-133 Section 6.2, key generation is performed as per FIPS 186-4; this is an allowed method as per FIPS 140-2 IG D.9
   - Establishment: N/A
   - Entry: N/A

- Output: Plaintext during TLS handshake
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 System key
- Key-to-Entity: This key belongs to the node
- Zeroization: N/A

6. Public key of an outside entity/server (Another SDKMS node)

- Description: 2048-bit RSA Key
- Generation: N/A - Generated outside of the module
- Establishment: N/A
- Entry: Plaintext during TLS handshake
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: This key belongs to an outside entity / server (Another SDKMS node)
- Zeroization: N/A

7. Public key of an outside entity/server (CA)

- Description: 2048-bit to 8192-bit RSA Key
- Generation: N/A - Generated outside of the module
- Establishment: N/A
- Entry: Plaintext during TLS handshake or during "Cluster Management" service
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: This key belongs to an outside entity/server (CA)
- Zeroization: N/A

8. RSA Public key of external Application

- Description: 2048-bit to 8192-bit RSA Key used for authentication using digital certificate
- Generation: N/A - Generated outside of the module
- Establishment: N/A
- Entry: Plaintext during "Authentication" service
- Output: N/A
- Storage: Plaintext in RAM
- Key-to-Entity: This key belongs to an outside entity, external Application
- Zeroization: N/A

9. User 2FA device public key

- Description: ECDSA P-256 key with SHA-256
- Generation: N/A - Generated outside of the module
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "System Configuration and management" service
- Output: N/A
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 System key
- Key-to-Entity: This key belongs to a specific user's two factor device
- Zeroization: N/A

10. SAML- Idp public key

- Description: 2048-bit RSA public key with SHA-256 provided as X.509 certificate.
- Generation: N/A - Generated outside of the module
- Establishment: N/A
- Entry: Automatic, encrypted over TLS session (with TLS encryption key (AES)) during "System Configuration and management" service
- Output: N/A
- Storage: Plaintext in RAM, encrypted in persistent storage with AES-GCM-256 System key
- Key-to-Entity: This key belongs to a specific account's SSO entry
- Zeroization: N/A

11. Appendix C: Acronyms

TERM	DESCRIPTION
AES	Advanced Encryption Standard (FIPS-197)
API	Application Programming Interface
CBC	Cipher Block Chaining
CTR	Counter
CO	Crypto Officer
DRBG	Deterministic Random Bit Generator (SP 800-90Ar1)
EMI/EMC	Electromagnetic Interference/Electromagnetic Compatibility
FIPS	Federal Information Processing Standards
FIPS 140-2 IG	Federal Information Processing Standards 140-2 Implementation Guidance

GCM	Galois/Counter Mode
HMAC	Keyed-hash Message Authentication Code (FIPS 198-1)
IV	Initialization Vector
KAT	Known Answer Test
N/A	Not Applicable
NDRNG	Non-deterministic random number generator
RAM	Random-access Memory
RBG	Random Bit Generator
RNG	Random Number Generator
SDKMS	Self-Defending Key Management Service™
SHA-1	Secure Hash Algorithm 1 (FIPS 180-4)
USB	Universal Serial Bus
VGA	Video Graphics Array

Table 17 Specification of acronyms and their descriptions

12. Appendix D: References

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

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

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

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

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

[SP 800-38F] Dworkin, Morris; Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST Special Publication 800-38D

[SP 800-56A] 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

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

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

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