# FIPS 140-2 Non-Proprietary Security Policy

---

## AWS Key Management Service HSM

(Hardware version 2.0, firmware version 1.3.6)

Document Version 1.00.14

August 24, 2017

# Table of Contents

# List of Tables



# List of Figures

# 1 Introduction

## 1.1 About FIPS 140

Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Communications Security Establishment Canada (CSE) Cryptographic Module Validation Program (CMVP) run the FIPS 140-2 program. The NVLAP accredits independent testing labs to perform FIPS 140-2 testing; the CMVP validates modules meeting FIPS 140-2 validation. *Validated* is the term given to a module that is documented and tested against the FIPS 140-2 criteria.

More information is available on the CMVP website at http://csrc.nist.gov/groups/STM/cmvp/index.html.

## 1.2 About this Document

This non-proprietary Cryptographic Module Security Policy for the AWS Key Management Service (KMS) Hardware Security Module (HSM) from Amazon Web Services (AWS) provides an overview of the HSM and a high-level description of how it meets the security requirements of FIPS 140-2. This document contains details on the module's cryptographic keys and critical security parameters. This Security Policy concludes with instructions and guidance on running the module in a FIPS 140-2 mode of operation.

AWS Key Management Service HSM may also be referred to as the "module" in this document.

The AWS Key Management Service HSM is used exclusively by AWS as a component of the AWS Key Management Service (KMS). The module is not directly accessible to customers of KMS. The cryptographic functions of the module are used to fulfill requests under specific public AWS KMS APIs.

## 1.3 External Resources

The AWS website (http://aws.amazon.com/kms/) contains information on AWS services that utilizes the module. The list of public AWS KMS APIs are documented on the AWS documentation website (http://docs.aws.amazon.com/kms/latest/APIReference/Welcome.html).

The Cryptographic Module Validation Program website contains links to the FIPS 140-2 certificate and AWS contact information.

## 1.4 Notices

This document may be freely reproduced and distributed in its entirety without modification.

## 1.5 Acronyms

Table 1 defines acronyms found in this document:

| Acronym | Term |
|---|---|
| AES | Advanced Encryption Standard |
| ANSI | American National Standards Institute |
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| CMK | Customer Master Key |
| CMVP | Cryptographic Module Validation Program |
| CO | Crypto Officer |
| CSE | Communications Security Establishment Canada |
| CSP | Critical Security Parameter |
| DH | Diffie-Hellman |
| DRBG | Deterministic Random Bit Generator |
| EC | Elliptic Curve |
| EMC | Electromagnetic Compatibility |
| EMI | Electromagnetic Interference |
| FCC | Federal Communications Commission |
| FIPS | Federal Information Processing Standard |
| HMAC | (Keyed-) Hash Message Authentication Code |
| HSM | Hardware Security Module |
| IPMI | Intelligent Platform Management Interface |
| KAS | Key Agreement Scheme |
| KAT | Known Answer Test |
| KMS | Key Management Service |
| MAC | Message Authentication Code |
| MD | Message Digest |
| NDRNG | Non Deterministic Random Number Generator |
| NIST | National Institute of Standards and Technology |
| NMI | Non-Maskable Interrupt |
| PKCS | Public-Key Cryptography Standards |
| PSS | Probabilistic Signature Scheme |
| RNG | Random Number Generator |
| RSA | Rivest, Shamir, and Adleman |
| SHA | Secure Hash Algorithm |

**Table 1 – Acronyms and Terms**

# 2 AWS Key Management Service HSM

## 2.1 Cryptographic Module Specification

AWS customers can use the AWS Key Management Service to generate and manage cryptographic keys and operate as a cryptographic service provider for protecting data within AWS. The AWS Key Management Service HSM provides dedicated cryptographic functions for the AWS Key Management Service.

The module runs firmware version 1.3.6 on hardware version 2.0 and is classified as a multi-chip standalone cryptographic module. The physical cryptographic boundary is defined as the module case, and the module runs on a non-modifiable operating environment.

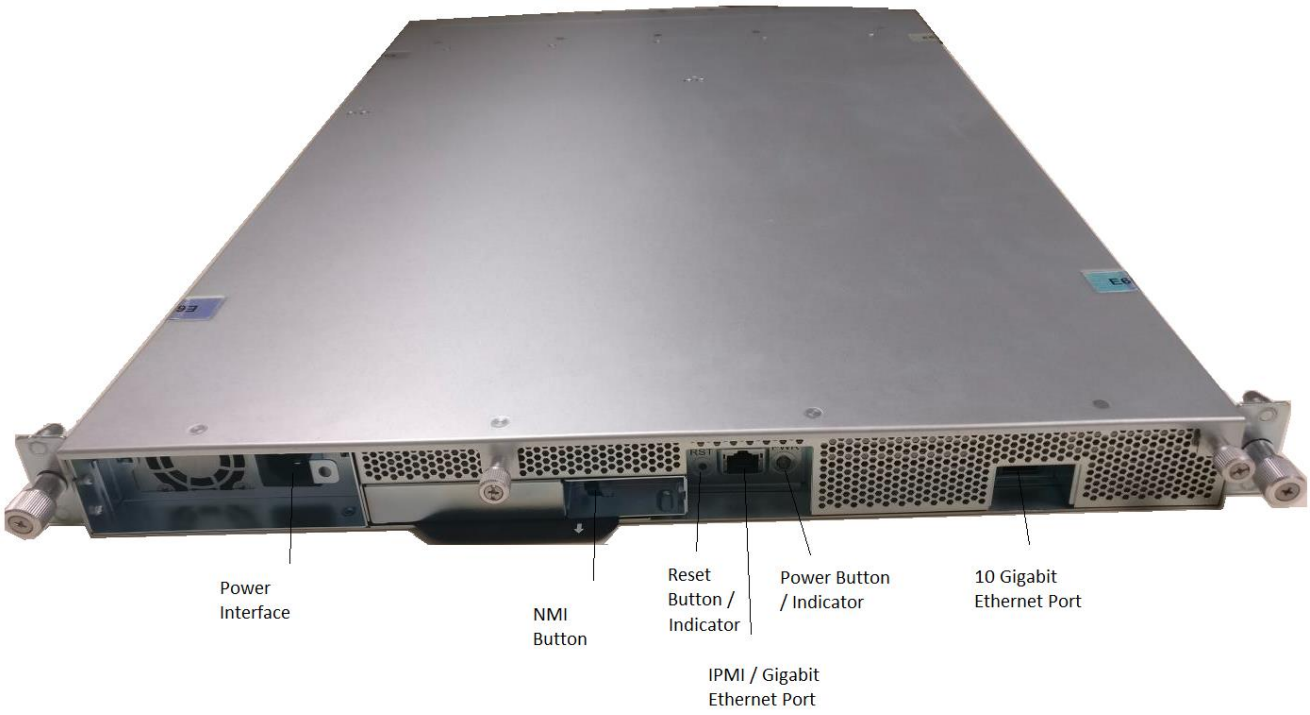


**Figure 1 – Cryptographic Module Boundary**

### 2.1.1 Validation Level Detail

Table 2 lists the level of validation for each area in FIPS 140-2:

| FIPS 140-2 Section Title | Validation Level |
|---|---|
| Cryptographic Module Specification | 3 |
| Cryptographic Module Ports and Interfaces | 2 |
| Roles, Services, and Authentication | 3 |
| Finite State Model | 2 |
| Physical Security | 3 |
| Operational Environment | N/A |
| Cryptographic Key Management | 2 |
| Electromagnetic Interference / Electromagnetic Compatibility | 2 |
| Self-Tests | 2 |
| Design Assurance | 3 |
| Mitigation of Other Attacks | N/A |
| **Overall Level** | **2** |

**Table 2 – Validation Level by FIPS 140-2 Section**

### 2.1.2 Approved Cryptographic Algorithms

The module's cryptographic algorithm implementations have received the following certificate numbers from the Cryptographic Algorithm Validation Program (CAVP). Although additional modes and key lengths were included in the CAVP algorithm testing, the table below represents the actual modes and key lengths used by the services of the module.

| CAVP Cert. | Algorithm | Standard | Mode/Method | Key Lengths, Curves or Moduli | Use |
|---|---|---|---|---|---|
| 4527 | **AES** | FIPS 197<br><br>SP 800-38A | ECB, CTR | 128, 256 | Encryption, Decryption |
| 4527 | **GCM/GMAC**[1] | SP 800-38D | AES | 128, 256 | Generation, Authentication, Encryption, Decryption |
| 4527 | **KTS** | SP 800-38F | AES GCM | 128, 256 | Key Transport |
| 1487 | **DRBG** | SP 800-90A | CTR DRBG | 256 | Random Bit Generation |
| 1102<br><br>1209 (CVL) | **ECDSA** | FIPS 186-4 | Signature Generation Component, Key Pair Generation, Signature Generation, Signature Verification, Public Key Validation | P-384 | Digital Signature Services |
| 2987 | **HMAC** | FIPS 198-1 | SHA-256 | 256 | Generation, Authentication |
| 122 | **KAS** | SP 800-56A | ED | P-384 | Key Agreement |
| 133 | **KBKDF, using Pseudorandom Functions** | SP 800-108 | Counter Mode | | HMAC-based KDF with SHA-256 |

[1] GCM with an internally generated IV. IV generation is compliant with IG A.5.

| CAVP Cert. | Algorithm | Standard | Mode/Method | Key Lengths, Curves or Moduli | Use |
|---|---|---|---|---|---|
| 2464<br><br>1208 (CVL) | **RSA** | FIPS 186-2<br><br>FIPS 186-4 | | 2048, 3072 and 4096 bits | Key Pair Generation, Signature Verification, Component Test |
| 3708 | **SHA** | FIPS 180-4 | SHA-256, SHA-384 | -- | Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications |

**Table 3 - FIPS-Approved Algorithms and Certificate Numbers**

The following Approved cryptographic algorithms were tested with vendor affirmation.

| Algorithm | IG Reference | Use |
|---|---|---|
| Key Transport Using RSA | Vendor Affirmed IG D.4 | [SP 800-56B, Section 9]<br>Optional RSA encapsulation scheme for protecting keys that customers import into AWS KMS.<br><br>RSA-OAEP with, and without key confirmation<br>Key sizes: 2048, 3072, and 4096 bits |

**Table 4 – Approved Cryptographic Functions Tested with Vendor Affirmation**

### 2.1.3 Non-Approved but Allowed Algorithms

The module supports the following non-FIPS 140-2 approved but allowed algorithms that may be used in the Approved mode of operation.

| Algorithm | Use |
|---|---|
| RSA Key Transport with PKCS #1 v1.5 | [IG D.9]<br>Optional RSA encapsulation scheme for protecting keys that customers import into AWS KMS.<br><br>Key sizes: 2048, 3072 and 4096 bits (key wrapping; key establishment methodology provides between 112 and 150 bits of encryption strength) |

| Algorithm | Use |
|---|---|
| EC Diffie-Hellman | [IG D.8]<br><br>Curves: NIST P-384 EC Diffie-Hellman (CVL Cert. 1209; key agreement; key establishment methodology provides 192 bits of encryption strength); |
| NDRNG | Provides seeding material for the DRBG. The NDRNG provides a minimum of 384 bits of entropy. |

**Table 5 – Non-Approved but Allowed Cryptographic Algorithms**

## 2.2 Module Interfaces

Table 6 describes the main interfaces of the module:

| Physical Interface | Description / Use |
|---|---|
| 10 Gigabit Ethernet Port | Main session interface for cryptographic services |
| IPMI / Gigabit Ethernet Port | Provides serial console access to query power on / off, control input, and status output. |
| Power Interface | Accept and provide power to the module |
| Power Button / Indicator | Turns the module on or off |
| Reset Button / Indicator | Restarts the module. The Reset Indicator is always on. |
| NMI Button / Indicator | Log hardware status information, such as fan speed and temperature to the IPMI system event log. The NMI Indicator is always on. |

**Table 6 – Interface Descriptions**

The module provides a number of physical and logical interfaces to the device, and the physical interfaces provided by the module are mapped to four FIPS 140-2 defined logical interfaces: data input, data output, control input, and status output. The logical interfaces and their mapping are provided in the following table:

| FIPS 140-2 Logical Interface | Module Physical Interface |
|---|---|
| Data Input | 10 Gigabit Ethernet Port |
| Data Output | 10 Gigabit Ethernet Port |
| Control Input | 10 Gigabit Ethernet Port<br>IPMI / Gigabit Ethernet Port<br>Power Button / Indicator<br>Reset Button / Indicator<br>NMI Button / Indicator |
| Status Output | 10 Gigabit Ethernet Port<br>IPMI / Gigabit Ethernet Port<br>Power Button / Indicator<br>Reset Button / Indicator<br>NMI Button / Indicator |
| Power | Power Plug |

**Table 7 – Logical Interface / Physical Interface Mapping**

## 2.3 Roles, Services, and Authentication

Operators of the module may assume the following three roles:

| Roles | Description | Authentication |
|---|---|---|
| **KMS Front End Role (KMS-FE)** | The KMS front end hosts perform actions on behalf of customers of AWS KMS. | RSA 2048, 4096 or ECDSA P384 |
| **KMS Coordinator Role (KMS-C)** | Non-public facing KMS hosts perform actions on behalf of KMS administrators in the Administrator Role. | RSA 2048, 4096 or ECDSA P384 |
| **Administrator Role (Admin)** | Employees of AWS who are authorized to manage the module. | RSA 2048, 4096 or ECDSA P384 |

**Table 8 – Roles and Authentication**

For FIPS purposes, the KMS Coordinator and Administrator roles serve as the Cryptographic Officer role per FIPS 140-2 requirements. The KMS-Front End role serves as the User role per FIPS 140-2 requirements.

The module supports identity-based authentication, and the respective services for each role are described in the following sections. The module does not support a Maintenance role.

Services supported by the module may also be referred to as APIs in this document.

The module supports authentication using RSA with 2048 and 4096 bit keys, and ECDSA with P-384. Operators of the module are identified by unique Operator Signature Public Key (QOS). The list of operator keys and the role of each operator are configured using either the Initialize or InitializeAndCreateDomain service. Operators interact with the module by submitting digitally sign commands to the module. The module authenticates operators by verifying the digitally signed commands submitted to the module.

The list of services supported by the module are listed in Table 9, Table 10, and Table 11. Unless otherwise specified, access to services can be configured to require one or more members of one or more roles listed in Table 8. These services are used only by components of KMS to fulfill requests under specific public AWS KMS APIs and cannot be used directly by KMS customers. See http://docs.aws.amazon.com/kms/latest/APIReference/Welcome.html for a list of the current public AWS KMS APIs.

### 2.3.1 Strength of Authentication

Authentication to the module requires RSA or ECDSA signature verification. These authentication methods are cryptographically strong. The possibility of a single random authentication attempt succeeding is far less than the required minimum of less than 1/1,000,000.

The possibility of a random authentication succeeding within a one-minute period is significantly less than 1/100,000. The cryptographic strengths of the digital signatures used for authentication create

such difficulty in achieving a successful random authentication attempt that even the theoretical maximum bandwidth of the 10 Gb/second Ethernet port is not significant to allow enough attempts in a one-minute period.

### 2.3.2 Cryptographic Services and Descriptions

For all cryptographic services in this section, all key/CSP input and output are encrypted using the HSM-to-Operator Session Key (HOSK) using 128 bit AES GCM. The use of the HOSK provides transport security between the HSM and other KMS Operators (as defined in section 2.3 above).

| HSM Service (API) | Roles | Description |
|---|---|---|
| Create | KMS Front End, KMS Coordinator, Administrators | Generates and encrypts either a HSM Backing Key (HBK) or an Import Wrapping Key (IWK) private key.<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** The Create API returns either:<br>• A HSM Backing Key encrypted with the active Domain Key ($DK_n$), or<br>• An Import Wrapping Key (IWK) key pair.<br>a. The IWK private key is encrypted with the active Domain Key ($DK_n$).<br>b. The IWK public key.<br><br>**Key/CSP Generated:**<br>• HSM Backing Key<br>• IWK public and private keys<br><br>**Key/CSP Read Access:**<br>• Active Domain Key ($DK_n$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The Create API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| ImportKey | KMS Front End, KMS Coordinator, Administrators | Decrypts a Customer Supplied Key (CSK) and re-encrypts it with the active Domain Key ($DK_n$)<br><br>**Key/CSP Input:**<br>• The private key of an Import Wrapping Key Pair (IWK) encrypted with the active or a recent iteration of domain key ($DK_n$ or $DK_{n-1}$)<br>• Customer Supplied Key (CSK), encrypted with the public key of the Import Wrapping Key.<br><br>**Key/CSP Output:** The Customer Supplied Key, encrypted with the current active domain key ($DK_n$)<br><br>**Key/CSP Read Access:** |

| HSM Service (API) | Roles | Description |
|---|---|---|
| | | • Active or a recent iteration of Domain Key ($DK_n$ or $DK_{n-1}$) used to encrypt the IWK private key<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The ImportKey API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| RefreshKey | KMS Front End, KMS Coordinator, Administrators | Re-encrypts a HSM Backing Key (HBK) key or Customer Supplied Key (CSK) encrypted with a recent iteration of the domain key ($DK_{n-1}$) with the active domain key ($DK_n$)<br><br>**Key/CSP Input:** HBK or CSK encrypted with a recent iteration of a Domain Key ($DK_{n-1}$)<br><br>**Key/CSP Output:** HBK or CSK encrypted with the active domain key ($DK_n$)<br><br>**Key/CSP Read Access:**<br>• Active or a recent iteration of Domain Key ($DK_n$ or $DK_{n-1}$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The RefreshKey API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| Encrypt | KMS Front End, KMS Coordinator, Administrators | Encrypt an arbitrary set of bytes using a key derived from the provided HBK or CSK.<br><br>**Key/CSP Input:** A HBK or CSK encrypted with the active or a recent iteration of domain key ($DK_n$ or $DK_{n-1}$)<br><br>**Key/CSP Output:** N/A (encrypted ciphertext)<br><br>**Key/CSP Read Access:**<br>• Active or a recent iteration of Domain Key ($DK_n$ or $DK_{n-1}$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The Encrypt API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| Decrypt | KMS Front End, KMS Coordinator, Administrators | Decrypts ciphertext using a key derived from the provided HBK or CSK.<br><br>**Key/CSP Input:**<br>• A HBK or CSK encrypted with a Domain Key ($DK_n$) |

| HSM Service (API) | Roles | Description |
|---|---|---|
| | | • Ciphertext or encrypted Customer Data Key (CDK)<br><br>**Key/CSP Output:** Arbitrary data or CDK encrypted using the HOSK<br><br>**Key/CSP Read Access:**<br>• Active or a recent iteration of domain key ($DK_n$ or $DK_{n-1}$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The Decrypt API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| Sign | KMS Front End, KMS Coordinator, Administrators | Performs a HMAC-SHA256 operation using the provided HBK or CSK<br><br>**Key/CSP Input:** HBK or CSK encrypted with the active domain key ($DK_n$)<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:**<br>• Active or a recent iteration of domain key ($DK_n$ or $DK_{n-1}$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The Sign API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| Verify | KMS Front End, KMS Coordinator, Administrators | Performs a HMAC-SHA256 verification operation using the provided HBK or CSK<br><br>**Key/CSP Input:** HBK or CSK encrypted with the active domain key ($DK_n$)<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:**<br>• Active or a recent iteration of domain key ($DK_n$ or $DK_{n-1}$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The Verify API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| EncryptRandomBytes | KMS Front End, KMS Coordinator, Administrators | Generate a number of random bytes and encrypt it using a key derived from the specified HBK or CSK. The random bytes may be used as cryptographic key material as Customer Data Keys (CDK).<br><br>**Key/CSP Input:** HBK or CSK encrypted by the active domain key ($DK_n$) |

| HSM Service (API) | Roles | Description |
|---|---|---|
| | | **Key/CSP Output:** A number of random bytes that may be used as Customer Data Keys (CDK) encrypted by the HBK or CSK.<br><br>**Key/CSP Read Access:**<br>• Active or a recent iteration of domain key ($DK_n$ or $DK_{n-1}$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The EncryptRandomBytes API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| GenerateAndEncryptRandomBytes | KMS Front End, KMS Coordinator, Administrators | Generate a number of random bytes for use and encrypt it using a key derived from the specified HBK or CSK. The random bytes may be used as cryptographic key material as Customer Data Keys (CDK). Note that the GenerateAndEncryptRandomBytes API will return encrypted versions of the random bytes in 2 forms.<br><br>**Key/CSP Input:** HBK or CSK encrypted by the active domain key ($DK_n$)<br><br>**Key/CSP Output:**<br>• A number of random bytes that may be used as Customer Data Keys (CDK) encrypted by the HOSK<br>• A number of random bytes that may be used as Customer Data Keys (CDK) encrypted by the HBK or CSK.<br><br>**Key/CSP Read Access:**<br>• Active or a recent iteration of domain key ($DK_n$ or $DK_{n-1}$)<br>• HSM-to-Operator Session Key (HOSK)<br><br>**Additional Information:** The GenerateAndEncryptRandomBytes API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |
| Generate | KMS Front End, KMS Coordinator, Administrators | Generate a specified number of random bytes, up to 1024 bytes.<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:**<br>• HSM-to-Operator Session Key (HOSK) |

| HSM Service (API) | Roles | Description |
|---|---|---|
| | | **Additional Information:** The Generate API validates the HSM-to-Operator Session Key (HOSK) to authenticate the call originates from an authenticated operator. The HOSK is also used to encrypt all input and output parameters. |

**Table 9 - Cryptographic Services and Descriptions**

### 2.3.3 Configuration Services and Descriptions

| HSM Service (API) | Roles | Description |
|---|---|---|
| CreateDomain | KMS Front End, KMS Coordinator, Administrators | Creates a new domain token for a new domain, but does not join the HSM to the domain yet.<br><br>**Key/CSP Input:**<br>• List of Operator Signature Public Keys (QOS)<br><br>**Key/CSP Generated:**<br>• HSM Signature Key Pair (HSK)<br>• HSM Agreement Key Pair (HAK)<br>• Initial Domain Key ($DK_0$)<br><br>**Key/CSP Output:** A Domain Token containing:<br>• List of Operator Signature Public Keys (QOS)<br>• List of HSM Signature Public Keys (QHSK) of all members of the domain<br>• List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>• Encrypted Initial Domain Key ($DK_0$)<br>• Encrypted Domain Key Encryption Key (DKEK)<br><br>**Key/CSP Read Access:** None |
| IngestDomain | KMS Front End, KMS Coordinator, Administrators | Joins a domain or receive an updated domain token.<br><br>**Key/CSP Input: A** Domain Token containing the following CSPs:<br><br>• List of Operator Signature Public Keys (QOS)<br>• List of HSM Signature Public Keys (QHSK) of all members of the domain<br>• List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>• Encrypted Domain Keys ($DK_n$)<br>• Encrypted Domain Key Encryption Key (DKEK).<br><br>**Key/CSP Generated:** None<br><br>**Key/CSP Output:** The unmodified input Domain Token<br><br>**Key/CSP Read Access:** |

| HSM Service (API) | Roles | Description |
|---|---|---|
| | | • HSM Signature Public Key (QHSK) of a known member of the domain<br>• HSM Agreement Private Key (dHAK)<br>• Operator Signature Public Keys (QOS)<br><br>**Key/CSP Write Access:**<br>• Domain Key ($DK_n$)<br>• Operator Signature Public Keys (QOS)<br>• HSM Signature Public Keys (QHSK) of all members of the domain<br>• HSM Key Agreement Public Keys (QHAK) of all members of the domain<br><br>**Additional Information:** When using the IngestDomain API to set up the first domain member, the operator(s) must meet the quorum configuration in the to-be-ingested domain. When using the IngestDomain API to ingest subsequent domains, the operator(s) must meet the quorum configuration in both the first domain, and in the to-be-ingested domain. |
| ForgetDomain | KMS Front End, KMS Coordinator, Administrators | Deletes domain information as it pertains to a particular domain on the module including all Domain Keys ($DK_n$, $DK_{n-1}$), effectively leaving the domain.<br><br>**Key/CSP Input: A** Domain Token containing the following CSPs:<br>• List of Operator Signature Public Keys (QOS)<br>• List of HSM Signature Public Keys (QHSK) of all members of the domain<br>• List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>• Encrypted Domain Keys ($DK_n$)<br>• Encrypted Domain Key Encryption Key (DKEK)<br><br>**Key/CSP Output:** The unmodified input Domain Token<br><br>**Key/CSP Read Access:**<br>• Domain Key ($DK_n$)<br>• Operator Signature Public Keys (QOS)<br><br>**Key/CSP De-Referenced:**<br>• Domain Key ($DK_n$)<br>• Operator Signature Public Keys (QOS)<br>• HSM Signature Public Keys (QHSK) of all members of the domain<br>• HSM Key Agreement Public Keys (QHAK) of all members of the domain |

| HSM Service (API) | Roles | Description |
|---|---|---|
| GetDomain | KMS Front End, KMS Coordinator, Administrators | Retrieves the current version of the domain token for a specified domain.<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** A Domain Token containing:<br>• List of Operator Signature Public Keys (QOS)<br>• List of HSM Signature Public Keys (QHSK) of all members of the domain<br>• List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>• Encrypted Domain Keys ($DK_n$)<br>• Encrypted Domain Key Encryption Key (DKEK)<br><br>**Key/CSP Read Access:**<br>• Domain Key ($DK_n$)<br>• Operator Signature Public Keys (QOS) |
| ChangeDomain | KMS Front End, KMS Coordinator, Administrators | Modifies the current state of an operational domain.<br><br>**Key/CSP Input:**<br>• A Domain Token containing:<br>  ○ List of Operator Signature Public Keys (QOS)<br>  ○ List of HSM Signature Public Keys (QHSK) of all members of the domain<br>  ○ List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>  ○ Encrypted Domain Keys ($DK_n$)<br>  ○ Encrypted Domain Key Encryption Key (DKEK)<br>• HSM Signature Public Keys (QHSK) and HSM Key Agreement Public Keys (QHAK) of the domain members to be added (optional)<br>• List of Operator Signature Public Keys (QOS) (optional)<br><br>**Key/CSP Generated:**<br>• Domain Key Encrypting Key (DKEK)<br>• HSM Ephemeral Agreement Key (dE, QE)<br><br>**Key/CSP Output:** An updated Domain Token containing the following CSPs:<br>• List of Operator Signature Public Keys (QOS)<br>• List of HSM Signature Public Keys (QHSK) of all members of the domain<br>• List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>• Encrypted Domain Keys ($DK_n$)<br>• Encrypted Domain Key Encryption Key (DKEK)<br><br>**Key/CSP Read Access:** Domain Key ($DK_n$), HSM Agreement Key (HAK), HSM Signature Key (HSK) |

| HSM Service (API) | Roles | Description |
|---|---|---|
| Initialize | All / unauthenticated | Initializes the HSM by generating the HSM Signature Key and HSM Agreement Key and configuring the HSM's operator and access control using a domain token from another HSM.<br><br>The Initialize API is only used during the module setup and initialization process. If the HSM is already initialized by a call to either the Initialize or InitializeAndCreateDomain API, the Initialize API will return an error as the HSM cannot be Initialized again without a reboot.<br><br>**Key/CSP Input:** One or more Domain Tokens. Each Domain Token contains:<br>• List of Operator Signature Public Keys (QOS)<br>• List of HSM Signature Public Keys (QHSK) of all members of the domain<br>• List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>• Encrypted Domain Keys ($DK_n$)<br>• Encrypted Domain Key Encryption Key (DKEK)<br><br>**Key/CSP Generated:**<br>• HSM Signature Key (HSK)<br>• HSM Agreement Key (HAK)<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:** None<br><br>**Key/CSP Write Access:**<br>• HSM Signature Key (HSK)<br>• HSM Agreement Key (HAK)<br>• Operator Signature Public Keys (QOS)<br><br>**Additional Information:** The Initialize API is unauthenticated. Initialize will fail if the HSM is already initialized by a call to either the Initialize or InitializeAndCreateDomain API. |
| InitializeAndCreateDomain | All / unauthenticated | Initializes the HSM by generating the HSM Signature Key and HSM Agreement Key, configuring the list of operators, roles and the quorum-based access control ruleset for all services / APIs.<br><br>The InitializeAndCreateDomain API is only used during the module setup and initialization process. If the HSM is already initialized by a call to either the Initialize or InitializeAndCreateDomain API, the InitializeAndCreateDomain API will return an error as the HSM cannot be Initialized again without a reboot.<br><br>**Key/CSP Input:**<br>• List of Operator Signature Public Keys (QOS) |

| HSM Service (API) | Roles | Description |
|---|---|---|
| | | **Key/CSP Generated:**<br>• HSM Signature Key Pair (HSK)<br>• HSM Agreement Key Pair (HAK)<br>• Initial Domain Key ($DK_0$)<br><br>**Key/CSP Output:** A Domain Token containing:<br>• List of Operator Signature Public Keys (QOS)<br>• List of HSM Signature Public Keys (QHSK) of all members of the domain<br>• List of HSM Key Agreement Public Keys (QHAK) of all members of the domain<br>• Encrypted Initial Domain Key ($DK_0$)<br>• Encrypted Domain Key Encryption Key (DKEK)<br><br>**Key/CSP Read Access:** None<br><br>**Key/CSP Write Access:**<br>• HSM Signature Key (HSK)<br>• HSM Agreement Key (HAK)<br>• List of Operator Signature Public Keys (QOS)<br><br>**Additional Information:** The InitializeAndCreateDomain API is unauthenticated. The InitializeAndCreateDomain API will fail if the HSM is already initialized by a call to either the Initialize or InitializeAndCreateDomain API. |
| Attest | KMS Front End, KMS Coordinator, Administrators | The Attest API is used by operators to attest an initialized HSM to ensure that the system is running the correct software, and to obtain an authentic copy of its credentials prior to being added to a domain<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:**<br>• HSM Signature Public Key (QHSK)<br>• HSM Agreement Public Key (QHAK)<br><br>**Key/CSP Read Access:**<br>• HSM Signature Key Pair (dHSK, QHSK)<br>• Host Agreement Public Key (QHAK)<br>• Operator Signature Public Key(s) (QOS) |

| HSM Service (API) | Roles | Description |
|---|---|---|
| Wipe | All / unauthenticated | The Wipe API will delete the HSM Signature Key and HSM Agreement Key from volatile memory. The Wipe API will fail unless all previously created domains in the module have been deleted using the ForgetDomain API.<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:** None<br><br>**Key/CSP De-Referenced:**<br>• HSM Signature Key Pair (HSK)<br>• HSM Agreement Key Pair (HAK)<br><br>**Additional Information:** This call is unauthenticated. |
| GetInitialDomainName | All / unauthenticated | Retrieves the initial domain name from an uninitialized HSM that is used as part of the domain creation bootstrap process.<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:** No access to Keys/CSPs.<br><br>**Additional Information:** This call is unauthenticated. |
| DeactivateAndReboot<br><br>(This service also performs the self-tests to run after the module is rebooted) | All / unauthenticated | The DeactivateAndReboot API returns the HSM to the factory state and reboots after verifying the HSM Signature Key and HSM Agreement Key have been deleted by the Wipe API.<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:** No access to Keys/CSPs.<br><br>**Additional Information:** This call is unauthenticated. |

| HSM Service (API) | Roles | Description |
|---|---|---|
| NegotiateSessionKey | One member from any role | Uses a set of identity keys to securely negotiate a session key that can be used between a KMS host and any HSM in the domain.<br><br>**Key/CSP Input:**<br>• Operator Ephemeral Agreement Public Key (QOEAK)<br><br>**Key/CSP Generated:**<br>• HSM Ephemeral Agreement Key Pair (dE, QE)<br>• HSM-Operator Session Key (HOSK)<br><br>**Key/CSP Output:**<br>• Encrypted HSM-Operator Session Key (HOSK)<br>• HSM Ephemeral Agreement Public Key (QE)<br><br>**Key/CSP Read Access:**<br>• Operator Signature Public Key (QOS)<br>• HSM Signature Key (dHSK)<br><br>**Key/CSP Write Access:**<br>• HSM-Operator Session Key (HOSK) |

**Table 10 - Configuration Services and Descriptions**

### 2.3.4 Audit Log Services and Descriptions

| Service (API) | Role | Description |
|---|---|---|
| ListLogs | KMS Front End, KMS Coordinator, Administrators | Returns a list of audit log file names.<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:** Operator Signature Public Key (QOS) |
| GetLog | KMS Front End, KMS Coordinator, Administrators | Retrieves specified audit log files<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:** Operator Signature Public Key (QOS) |

| Service (API) | Role | Description |
|---|---|---|
| DeleteLog | KMS Front End, KMS Coordinator, Administrators | Deletes specified audit log file<br><br>**Key/CSP Input:** None<br><br>**Key/CSP Output:** None<br><br>**Key/CSP Read Access:** Operator Signature Public Key (QOS) |

**Table 11 – Audit Log Services and Descriptions**

### 2.3.5 Show Status

The module supports the following APIs to return status information.

| Status Service (API) | Description |
|---|---|
| Ping | Returns “healthy” if the module is initialized and has ingested a domain<br><br>Returns “failure” otherwise |
| Fips | Returns “healthy” if the module is operating in FIPS mode<br><br>Returns “failure” if the module is not operating in FIPS mode |

**Table 12 - Status Services and Descriptions**

In addition, an operator with access to the serial console can obtain hardware status information such as temperature, fan speed, etc.

### 2.3.6 Zeroization

Zeroization is accomplished by powering off the module.

## 2.4 Physical Security

The module is a multiple-chip standalone module and conforms to Level 3 requirements for physical security. The module’s production-grade enclosure is made of a hard metal, and the enclosure contains a removable cover. The baffles installed by AWS satisfy FIPS 140-2 requirements for module opacity, and an internal tamper switch zeroizes CSPs at power on / power off when triggered, satisfying Level 3 requirements.

## 2.5 Operational Environment

The module operates in a non-modifiable operational environment.

The module meets Federal Communications Commission (FCC) FCC Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined by 47 Code of Federal Regulations, Part15, Subpart B.

## 2.6 Cryptographic Key Management

### 2.6.1 Critical Security Parameters

Table 13 provides a complete list of Critical Security Parameters used within the module. All keys and CSPs are zeroized by powering off the module.

| Keys and CSPs | Key Description | Algorithm and Key Size | Generation | Input / Output Method | Storage |
|---|---|---|---|---|---|
| HSM Backing Key (HBK) | HSM Backing Keys are a 256-bit master key, from which specific-use keys may be derived using the SP800-108 CTR key derivation function. The HBK or keys derived from the HBK are used to encrypt CDKs.<br><br>The HBK maps to the Customer Master Key construct exposed in the public AWS KMS API. | AES GCM 256 bits | Internally using DRBG, or imported from another member of a Domain | **Input:** Encrypted with the Domain Key using AES GCM.<br>**Output:** Encrypted with the Domain Key using AES GCM. | Volatile memory only |
| Customer Data Key (CDK) | Customer data keys are exported by the EncryptRandomBytes and GenerateAndEncryptRandomBytes API.<br><br>The use of CDKs are unknown to the module. The customer may obtain the CDK by sending the encrypted CDK to KMS to decrypt under an authenticated and authorized request. | N/A | Internally using DRBG or imported from another member of a Domain | **Input:** Encrypted with a key derived from a HBK or CSK using AES GCM.<br>**Output:** Encrypted in 2 forms by the GenerateAndEncryptRandomBytes API:<br>• Encrypted with a key derived from a HBK or CSK and<br>• Encrypted with the HOSK to provide secure transport to the requesting service operator/role.<br><br>EncryptRandomBytes exports the CDK encrypted with a key derived from a HBK or CSK. | Volatile memory only |

| Keys and CSPs | Key Description | Algorithm and Key Size | Generation | Input / Output Method | Storage |
|---|---|---|---|---|---|
| Domain Key (DK) | A Domain Key is shared among all the members of a Domain and is used to encrypt HBKs, CSKs, and host-operator session keys (HOSKs). | AES GCM 256 bits | Internally using DRBG, or imported from another member of a Domain | **Input:** DK encrypted with the DKEK may be imported to other members of a Domain **Output:** DK encrypted with the DKEK may be exported to other members of a Domain | Volatile memory only |
| Domain Key Encryption Key (DKEK) | A Domain Key Encryption Key is generated on a Host and is used for encrypting the current set of domain keys when sharing of the domain state between HSM hosts. | AES GCM 256 bits | Internally using DRBG or externally by another member of a Domain | **Input:** The DKEK is encrypted with the shared secret generated from the HSM's Key Agreement Key (QHAK) and another HSM's Ephemeral Key Agreement Key (dE). **Output:** The DKEK is encrypted with the shared secret generated from the HSM's Key Agreement Key (dHAK) and another HSM's Ephemeral Key Agreement Key (QE). | Volatile memory only |
| HSM Agreement Key Pair (dHAK, QHAK) | Every initialized HSM has a locally generated Elliptic Curve Diffie-Hellman agreement key pair used to encrypt/decrypt DKEKs between HSMs. | Elliptic Curve Diffie-Hellman agreement key pair on the curve secp384r1 (NIST-P384) | Internally using DRBG | **Input: N/A** **Output:** The public key (QHAK) is exported in plaintext | Volatile memory only |
| HSM Ephemeral Agreement Key Pair (dE, QE) | These keys are generated in two cases: (i) to establish a HSM-to-HSM encryption key to transport DKEKs in domain tokens; (ii) to establish HSM-to-operator session keys to protect sensitive HSM-operator communications. | Elliptic curve Diffie-Hellman keys on the curve secp384r1 (NIST-P384) | Internally using DRBG | **Input:** N/A **Output:** The public key (QE) is exported in plaintext | Volatile memory only |
| HSM Signature Key Pair (dHSK, QHSK) | Every initiated HSM has a locally generated Elliptic Curve Signature key pair used to sign data created on the HSM. | Elliptic Curve Signature key pair on the curve secp384r1 (NIST-P384) | Internally using DRBG | **Input:** N/A **Output:** The public key (QHSK) is exported in plaintext | Volatile memory only |

| Keys and CSPs | Key Description | Algorithm and Key Size | Generation | Input / Output Method | Storage |
|---|---|---|---|---|---|
| HSM-Operator Session Key (HOSK) | Operator services establish an AES-128-GCM session key with the HSM to protect communication between operator services and HSMs in the same domain. | AES GCM 128-bits | Internally using DRBG, or imported from an HSM that is a member of the same domain | **Input:** The HOSK is encrypted with the domain key (DK).<br>**Output:**<br>The HOSK is encrypted with the domain key (DK) and an elliptic curve Diffie Hellman key exchange (NIST-P384) using the HSM Ephemeral Agreement Key (QE) and the Operator Ephemeral Agreement Public Key (QOEAK). | Volatile memory only |
| Import Wrapping Key (dIWK, QIWK) | The public key is used by customers of KMS to wrap their CSK for import via the public AWS KMS API. | RSA 2048, 3072, and 4096 bits | Internally using DRBG or imported from another member of a Domain | **Input:** The private key (dIWK) is encrypted with the Domain Key (DK) using AES-GCM.<br>**Output:** the private key (dIWK) is encrypted with the Domain Key (DK) using AES-GCM. The public key (QIWK) is exported in plaintext. | Volatile memory only |
| Customer Supplied Key (CSK) | Key generated by a customer of KMS outside the AWS KMS system.<br><br>CSK is a 256-bit master key, from which specific-use keys may be derived using the SP800-108 CTR key derivation function. The keys derived from the CSK are used to encrypt CDKs.<br><br>The CSK maps to the Customer Master Key construct exposed in the public AWS KMS API. | AES GCM 256 bits | Externally by AWS KMS customers | **Input:** CSK is encrypted using Import Wrapping Key (QIWK) when used with the ImportKey API when the customer imports the key into the AWS KMS system. After import, the CSK is encrypted with the Domain Key using AES GCM.<br>**Output:** CSK encrypted by a Domain Key (DK). | Volatile memory only |
| DRBG (CTR AES) | Entropy input (length dependent on security strength) | SP 800-90A CTR DRBG V (128 bits) AES key ( 256) | Internally by NDRNG | **Input:** Directly from the internal NDRNG<br>**Output:** N/A | Volatile memory only |

**Table 13 – Module Keys/CSPs**

### 2.6.2 Public Keys

Table 14 shows the list of Public Keys used within the module with associated private keys that only exist outside of the module. All Public Keys are generated outside of the module.

| Public Key | Key Description | Algorithm and Key Size | Input / Output Method | Storage |
|---|---|---|---|---|
| Operator Ephemeral Agreement Public Key (QOEAK) | Operators establish a session key (HSM-Operator Session Keys) using an Elliptic Curve Diffie-Hellman key exchange on the curve secp384r1 (NIST-P384). | Elliptic Curve Diffie-Hellman (EC DH) ephemeral key agreement on the curve secp384r1 (NIST-P384) | **Input:** When an operator calls the NegotiateSessionKey service.<br>**Output:** N/A | Volatile memory |
| Operator Signature Public Key (QOS) | Both service operators and human operators have an identity signing key used to authenticate to the HSM. | Elliptic Curve Signature (EC DSA) on the curve secp384r1 (NIST-P384), RSA 2048, or RSA 4096 bits | **Input:** The public key (QOS) is imported in plaintext when an administrator calls InitializeAndCreateDomain, CreateDomain, and ChangeDomain. They are also imported by APIs that accepts a Domain Token.<br>**Output:** The public keys are exported from the HSM in plaintext by APIs that exports a Domain Token. | Volatile memory |

**Table 14 – Public Keys**

## 2.7 Self-Tests

FIPS 140-2 requires the module to perform self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. Some functions require conditional tests during normal operation of the module. All of these tests are listed and described in this section. In the event of a self-test error, the module will log the error and enter the error state. Once in the error state, all CSPs are zeroized and the module becomes unusable.

### 2.7.1 Power-On Self-Tests

Power-on self-tests are run upon the initialization of the module and do not require operator intervention to run. If any of the tests fail, the module will not initialize. The module will enter an error state and no services can be accessed by the operator. The module implements the following power-on self-tests:

| Type | Test |
|---|---|
| Integrity Check | • 160 bit error detection code (EDC) on all module components |

| Type | Test |
|---|---|
| Known Answer Tests | • AES (Encryption and decryption in ECB mode. Key size: 128 bit)<br>• AES GCM / GMAC (Generation and verification. Key size: 128 bit)<br>• ECC KAS (ECDH) (Primitive Z test. Parameter set: EC)<br>• ECDSA (signature generation and verification. Curve: P-256)<br>• RSA (Signature generation and verification, key transport SP800-56B per IG 9.4)<br>• HMAC (Generation and verification with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)<br>• SHS (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, verified as part the respective HMAC tests)<br>• SP 800-90 CTR_DRBG |

**Table 15 – Power-On Self-Tests**

Each module performs all power-on self-tests automatically when the module is initialized. All power-on self-tests must be passed before a User/Crypto Officer can perform services. The Power-on self-tests can be run on demand by rebooting the module in FIPS approved Mode of Operation.

### 2.7.2 Conditional Self-Tests

Conditional self-tests are test that run during operation of the module. If any of these tests fail, the module will enter an error state, where no services can be accessed by the operators. The module can be re-initialized to clear the error and resume FIPS mode of operation. Each module performs the following conditional self-tests:

| Type | Description |
|---|---|
| Pair-wise Consistency Tests | • RSA key pair generation<br>• ECDSA key pair generation |
| SP 800-56A Assurances | • Performed per SP 800-56A Sections 5.5.2 and 5.6.2. Required per IG 9.6 |
| Continuous RNG Tests | • Performed on NDRNG per IG 9.8<br>• SP 800-90 CTR_DRBG |
| DRBG Health Tests | • Performed on DRBG, per SP 800-90A Section 11.3. Required per IG C.1. |

**Table 16 – Conditional Self-Tests**

The module does not perform a firmware load test because no additional firmware can be loaded in the module while operating in FIPS-approved mode. Please see Section 3 for guidance on configuring and maintaining FIPS mode.

### 2.7.3 On-demand Self-Tests

On-demand self-tests can be performed by rebooting the module which will perform the power-on self-tests as described in 2.7.1.

## 2.8 Mitigation of Other Attacks

The module does not mitigate other attacks.

# 3 Guidance and Secure Operation

The module only supports FIPS-mode of operation. Beyond initial setup, no specific technical steps are required to configure FIPS-mode of operation.

## 3.1 Crypto Officer Guidance

Only authorized AWS employees may assume the Administrator (Crypto Officer) role.

The following section provides a high-level overview to configure the HSM. Members of the Administrator role (Crypto Officer) must follow the AWS internal guidance published in the Operation Guidance for the AWS Key Management Service.

### 3.1.1 Module Inspection

Tamper evidence labels on the module must be inspected to verify that no attempts have been made to open the module.

The module must be inspected upon initial delivery and after the module reboots due to unscheduled/unexpected power events.

If evidence of a tamper is discovered, the module shall be removed from operation immediately. If new seals are required, the module shall be replaced with a new module with factory-applied seals

Figure 2 identifies the locations of the four tamper evidence labels applied during the manufacturing process.

**Figure 2 – Tamper Evidence Label Locations**

### 3.1.2 Initial Configuration

When setting up the first HSM member of a new domain, call the InitializeAndCreateDomain service with the list of operators' Operator Signature Public Keys, their respective roles and the access policy for each service (API) in accordance with AWS internal guidance and procedures.

Services / APIs that modifies an HSM's domain membership or configuration must be configured to require a quorum of two Crypto Officers.

When setting up subsequent members of an existing domain, the administrator first retrieves domain information from an existing domain member using the GetDomain service. The Initialize service can then be used to initialize the new HSM with the configuration of the existing domain.

Ensure each HSM is operating in FIPS mode by calling the Fips status API.

## 3.2 User Guidance

### 3.2.1 General Guidance

No additional guidance is required to maintain FIPS mode of operation. The only users of the HSM are the front end hosts of the AWS Key Management Service.