# Cygnus X3 Hardware Security Module (XHSM) Security Policy

KMS 6
Mailing Solutions Management Engineering
Version 01.08
12/11/12


Pitney Bowes Inc
37 Executive Drive
Danbury, CT 06810

# Contents

# 1. Module Overview

This document describes the security policy for the Pitney Bowes Cygnus X3 Hardware Security Module (XHSM) Cryptographic Module.

| Item | Version |
|---|---|
| Hardware | P/N 1R84000 Version A |
| Firmware | 01.00.06 |
| Device Abstraction Layer (DAL) (includes SW Download Utility) | 01.03.0074 |

***Cryptographic systems rely on the confidentiality of private and secret keys. An HSM provides a protected environment where cryptographic operations can be performed using public, private or secret keys. The Cygnus XHSM cryptographic module is a single-chip module. The module's cryptographic boundary is defined as the package of the secure processor, the Sigma ASIC, designed by Pitney Bowes.***

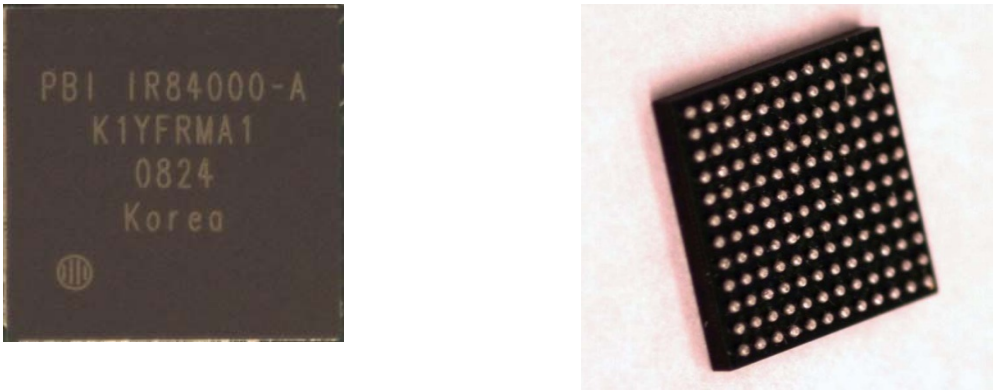


*Figure 1 – Cygnus XHSM Cryptographic Module (Sigma ASIC Secure Processor)*

# 2. Security Level

The Cygnus XHSM cryptographic module meets the overall requirements applicable to Level 3 security of FIPS 140-2.

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

*Figure 2 - Module Security Level Specification*

# 3. Modes of Operation

The XHSM operates in three modes

1. Operational Mode – FIPS compliant
2. Manufacturing Mode (out of scope for FIPS validation)
3. Custom Application Mode (out of scope for FIPS validation)

The XHSM mode is determined by sending the HSMGetStatus command. Once in Operational Mode, the module is always operating in an Approved mode of operation as defined by FIPS 140-2.

The module supports the following FIPS Approved algorithms:

| Algorithm | Usage |
|---|---|
| FIPS 186-3 DSA (Cert. #632) | This algorithm is used to generate key pairs, digitally sign and verify signatures according to FIPS186-3 for L=1024, N=160 & SHA-1. |
| FIPS 186-3 ECDSA (Cert. #286) | This algorithm is used to generate key pairs, digitally sign and verify signatures according to FIPS186-3 for P curves 192 (SHA-1) and 256 (SHA-256). Key Pair generation per FIPS 186-3 Section B.4.2. |

| Algorithm | Usage |
|---|---|
| SHA-1 & SHA-256 (Cert. #1733) | SHA-1 provides the hashing algorithm used as part of the digital signature process for DSA and ECDSA and in the generation of SHA-1 HMAC. SHA-256 provides the hashing algorithm used as part of the digital signature process for ECDSA and in the generation of SHA-256 HMAC. |
| AES CBC (Cert. #1979) | This encryption algorithm is used to encrypt and decrypt other cryptographic information for secure delivery. Key sizes supported are 128 bits and 256 bits. |
| AES ECB (Cert. #1979) | This encryption algorithm is used as a part of the AES Key Wrap. Key sizes supported are 128 bits and 256 bits. |
| Hash DRBG (Cert. #181) | SP 800-90 Hash-Based Deterministic Random Bit Generator using SHA-256. |
| ECCDH Primitive (CVL Cert. #20) | KAS and KDF per SP 800-56A used in establishing a session key. Supports P curve: 256 bits |
| HMAC (Cert. #1192) | Used to generated Message Authentication Codes |
| KAS (Cert. #33) | Key Agreement Protocol used to establish a session key |
| Triple-DES CBC (Cert. #1319) | Legacy encryption support 3DES2 and 3DES3 (EDE and EEE) keys |
| Triple-DES ECB (Cert. #1319) | Legacy encryption support 3DES2 and 3DES3 (EDE and EEE) keys |
| Triple-DES MAC (Cert. #1319, vendor affirmed) | Legacy message authentication |
| FIPS 186-2 RSA PKCS 1.5 (Cert. #1063) | This algorithm is used to digitally sign and verify signatures according to FIPS186-2 for 1024 bit modulus using SHA-1. |
| FIPS 186-2 RSA X9.31 (Cert. #1063) | This algorithm is used to generate key pairs, digitally sign and verify signatures according to FIPS186-2 for 1024 bit modulus using SHA-1. |

The module supports the following non-Approved but Allowed security functions:

| | |
|---|---|
| AES Key Wrap (Cert. #1979) | Used to encrypt symmetric and private keys loaded into the HSM. Key sizes supported are 128 bits and 256 bits. (key wrapping; key establishment methodology provides 128 and 256 bits of encryption strength) |
| Diffie Hellman (using 1024 or 2048 bit keys) | Key Agreement Protocol used to establish a session key. Key agreement; key establishment methodology provides 80 or 112 bits of encryption strength |

SHA-224 is supported by the cryptographic module, but is not available for use as the module is configured for the current validation.

# 4. Ports and Interfaces

The Sigma ASIC is implemented as a 144-pin BGA where all power input, data input, data output, control input, and status output interfaces are supported.

| Type | Pin |
|---|---|
| Data Input | A1, B1, C12, A12 |
| Data Output | A1, B1, D12, A12 |
| Status Output | A1, B1, D1, E1, F2, E12, F11 |
| Control Input | A1, B1, B11, C9, C7, D2, E3, F1, F2, F3, F4, M1, K6, M8, M12, L12, L11, H10, H9, G12, G11, F11, C11 |
| Power | B10, A10, C10, B9, A9, D9, D8, A8, E8, A7, D7, E7, F7, E6, C5, D5, A4, C2, C3, D3, D4, E2, E4, E5, F5, F6, G6, G4, G5, H4, J4, J5, H5, L6, J6, H6, H7, J7, L8, J8, L10, M11, K11, J12, J10, J11, J9, H8, H12, H11, G10, F12, G9, G8, G7, F9, F8, B12 |
| Disabled | A11, B8, C8, B7, C6, A6, B6, D6, A5, B5, B4, C4, B3, A3, B2, A2, G1, G2, G3, H1, H2, H3, J1, J2, J3, K1, K2, L1, M2, L2, M3, L3, K3, M4, L4, K4, M5, L5, K5, M6, M7, L7, K7, K8, L9, M9, K9, K10, M10, K12, F10, E11, E10, D11, E9, D10 |

*Figure 3 – Interface Table*

# 5. Identification and Authentication Policy

The module supports two roles, the Crypto-Officer (CO) and the Trusted User. All services described in Section 6 below are available to both the CO and Trusted User.

| ***Role*** | ***Authentication Method*** | ***Authentication Type*** |
|---|---|---|
| *Crypto-Officer* | *Knowledge of a Shared Secret* | *Identity-based* |
| *Trusted User* | *Knowledge of a Shared Secret* | *Identity-based* |

*Figure 4 – Roles and Authentication Type*

| ***Authentication Mechanism*** | ***Strength Mechanism*** |
|---|---|
| *Shared Secret* | *Authentication is based on knowledge of either a 256 bit AES key (DPK), or a 256 bit HMAC shared secret (DAK). Both provide 256 bits of security. The probability of a random attempt or false acceptance occurring is then 1 in 2^256, which is less than 1 in 1,000,000.*<br><br>*The module can execute at most 3,000 authentication attempts per second, based on processing limitations; therefore, the probability of success in a one minute period is 180,000 in 2^256. This is far less than one in 100,000.* |

*Figure 5 –Authentication Strength*

# 6. Access Control Policy

Each service described below is available to both the CO and Trusted User. (See also Section 7.2.)

**Crypto-Officer (CO) and Trusted User Services:**

- **Generate**: The Crypto Officer or Trusted User sends this block to instruct the XHSM to generate a Public/Private key pair or a Secret key. The message specifies the cryptographic algorithm and the parameters for use in the generation of the key(s).
- **Load Key**: The Crypto Officer or Trusted User sends this command to instruct the XHSM to load an encrypted key record for later use. The command specifies the storage type:
    - Volatile: Store in RAM, can be replaced if space is needed.
    - Sticky: Store in RAM, can NOT be replaced until it is deleted by the host.
    - Static: Store in NVM
- **Split Key**: The Crypto Officer or Trusted User sends this command to instruct the XHSM to divide a key into 2 or more parts.
- **Join Key**: The Crypto Officer or Trusted User sends this command to instruct the XHSM to assemble a key that has been previously split.
- **Export Key**: The Crypto Officer or Trusted User sends this command to have a key securely exported for storage / use in another location.
- **Encrypt**: The Crypto Officer or Trusted User sends this command to encrypt data.
- **Decrypt**: The Crypto Officer or Trusted User sends this command to decrypt data.
- **Firmware Update** is described in section 7.1 Firmware Update
- **Load Parameters**: The Crypto Officer or Trusted User sends this message to load a set of parameters into the HSM.
- **Derive Key**: The Crypto Officer or Trusted User sends this message to generate a key to be used by the host and the HSM to exchange secure information.
- **Decrypt Encrypt**: The Crypto Officer or Trusted User sends this message to allow encrypted data to be decrypted and re encrypted with another key.
- **Decrypt Compare:** The Crypto Officer or Trusted User sends this message to allow encrypted data to be decrypted and compared.
- **Sign**: The Crypto Officer or Trusted User sends this message to apply a cryptographic signature to a set of data.
- **Verify**: The Crypto Officer or Trusted User sends this message to verify a cryptographic signature on a set of data.
- **Modular Exponentiation**: The Crypto Officer or Trusted User sends this message to perform the computation: $a^b \bmod n$.
- **Set Counter**: The Crypto Officer or Trusted User sends this message to set an internal counter to a specific value.

**Unauthenticated Services:**
Miscellaneous functions that do not require XHSM authentication of the entity; These services are available to all roles.

- **Delete Key**: The Host sends this message to remove a key from the HSM.
- **Get Random Number**: The Host sends this message to get a random number from the HSM.
- **Hash**: The Host sends this message to generate a hash based on a set of data.
- **Get Key List**: Instructs the HSM to return a list of all active keys stored in the HSM.
- **Update counters:** Instructs the HSM to update specific internal counters
- **Get Counter Record:** Instructs the HSM to output a copy of the current values of the internal counters.
- **Get Parameters:** Instructs the HSM to retrieve parameter values from the HSM. The Host can request individual parameter IDs or all of the stored parameters in the HSM.
- **Perform Diagnostic Test:** Instructs the HSM to request that the Cygnus HSM perform one or more diagnostic tests.
- **Read Log:** Instructs the HSM to get Log Data. The number of available entries, the size of each entry, and the data contained in each entry will depend on the log that is being requested.
- **Read Mfg Data:** The Host sends this message to retrieve manufacturing specific data for the HSM.
- **GetTimeDrift:** Instructs the HSM to retrieve the drift adjustment.
- **SetTimeDrift:** Instructs the HSM to set the drift adjustment.
- **Clear Private Information:** Instructs the HSM to remove any private or secret clear text material residing in RAM.
- **Add Log Entry:** Instructs the HSM to add an entry to an internal log in the HSM.
- **Delete Log:** Instructs the HSM to remove all entries of an internal log in the HSM.
- **Get Status:** Instructs the HSM to request HSM status information.
- **Get HW Status:** Instructs the HSM to request the hardware specific status data of the HSM.
- **Reboot:** Instructs the HSM to reboot the PSD application.
- **Get Versions:** Instructs the HSM to get the versions of the components in the HSM
- **Reinit:** Instructs the XHSM to erase all NVM data except for HW Mfg Data and ‘persistent’ data (total device cycles, reinit count) and then invalidates the HSM DAL. This command zeroizes the Unique HSM Key Encryption Key and Unique HSM Key Authentication Key, which result in the loss of all other Private and Secret Keys. Used to ‘clean’ the HSM so it can be re-configured.

# 7. Firmware Update Access Control Policy

The HSM supports a secure firmware update process. The Software Download Utility within the Device Abstraction Layer is responsible for firmware updates.

## 7.1 Device Abstraction Layer Download

Boot loader loads DAL in chunks. Each chunk is verified with signed hash record containing ECDSA 256 signature. When SDU download is complete a hash is calculated and verified on the entire DAL firmware image.

## 7.2 HSM App Download and Upgrades

The DAL loads HSM app in chunks. Each chunk is verified with signed hash record containing ECDSA 256 signature. When HSM App download is complete, a hash is calculated and verified on the entire HSM Application firmware image and a signature is verified on the entire HSM Application image.

The Software Download Utility supports the following messages:

**Crypto- Officer/Trusted User:**

- Setup Download Data: The Host sends this signed record to make the Software Download Utility aware of the parameters of the software (application) to be downloaded. This message is signed by the SWAK. Receipt of this message triggers a transition to the state required to load chunk information.
- Setup Download Chunk: The Host sends this signed record to make the Software Download Utility aware of the parameters of the software (application) chunk to be sent in the following message. Receipt of this message triggers a transition to the state required to load the chunk. The Setup Download Chunk message is only valid if the DAL has received a valid Setup Download Data message.
- Download Chunk: This message contains the data referenced in the Setup Download Chunk message.

**Utility Functions**
The following utility functions are unauthenticated and intended to aid the host application in managing the software update process.

- Reboot: This function is used to invoke a reboot. It returns a 'Reboot' response message, waits until the transmit channel is idle then resets the ASIC.
- Initialize: This function writes 0's to the DAL Software Validity Flag, sends a response message, waits for until the transmit channel is idle and then resets the ASIC. The HSM transitions to the ROM Firmware State after completion of the reboot.
- Get Status: The Host device sends this message to the HSM to request the current status information.

# 8. Definition of Critical Security Parameters (CSPs)

There are 5 CSPs that are necessary for the HSM to function as a FIPS device. Other keys maybe loaded or generated by command from the user. These keys could include DSA, ECDSA, AES, RSA, TDES, HMAC, ECDH private / secret / public keys to meet the needs of the TU while using the HSM.

The first 5 entries in the following table describe the 5 necessary CSPs contained in the module:

| Key | Key Name | Description / Usage | Generation / Agreement | Storage | Entry / Output | Destruction |
|---|---|---|---|---|---|---|
| KEK | Unique HSM Key Encryption Key | AES256Key Encryption Key | Internally by FIPS approved DRBG | Clear text | Entry: N/A<br>Output: N/A | Zeroized on Tamper or Reinitialize or removal of all power |
| KAK | Unique HSM Key Authentication Key | HMAC256Key Authentication Key | Internally by FIPS approved DRBG | Clear text | Entry: N/A<br>Output: N/A | Zeroized on Tamper or Reinitialize or removal of all power |
| DAK | DAL Authentication Key | HMAC256 Key | Entered in factory environment | ciphertext | Entry: N/A<br>Output: Encrypted | Encrypting key zeroized on Tamper or Reinitialize or removal of all power |
| DPK | DAL Privacy Key | AES256 Key | Entered in factory environment | ciphertext | Entry: N/A<br>Output: Encrypted | Encrypting key zeroized on Tamper or Reinitialize or removal of all power |
| V | DRBG Seed | DRBG seed | Entered in factory environment | Ciphertext | Entry: N/A<br>Output: N/A | Encrypting key zeroized on Tamper or Reinitialize or removal of all power |
| AK | Authentication Key | HMAC128 or HMAC256 or DSA1024 or ECDSA160 or, ECDSA192 or ECDSA256 or RSA1024 or 3DES2 or 3DES3 Key | Internally by FIPS approved DRBG | ciphertext | Entry: Encrypted<br>Output: Encrypted | Encrypting key zeroized on Tamper or Reinitialize or removal of all power |

| Key | Key Name | Description / Usage | Generation / Agreement | Storage | Entry / Output | Destruction |
|---|---|---|---|---|---|---|
| PK | Privacy Key | AES128 or AES256 or RSA1024 or 3DES3 Key or 3DES2 Key | Internally by FIPS approved DRBG | ciphertext | Entry: Encrypted Output: Encrypted | Encrypting key zeroized on Tamper or Reinitialize or removal of all power |
| SS | Shared Secret | Used to derive Session Key (SK) | DH or Key Agreement | ciphertext | Entry: Encrypted Output: N/A | Encrypting key zeroized on Tamper or Reinitialize or removal of all power |
| SK | Session Key | DH ECDH AES128 or AES256 or 3DES3 Key or 3DES2 Key | Derived from Shared Secret | ciphertext | Entry: N/A Output: Encrypted | Encrypting key zeroized on Tamper or Reinitialize or removal of all power |

*Figure 6 – CSP Table*

The following table describes the public keys contained in the module:

| Key | Key Name | Description / Usage | Generation / Agreement | Storage | Entry / Output |
|---|---|---|---|---|---|
| SMAK | Domain Comet Authentication Sigma Manufacturing Key | ECDSA used to validate Software Download Utility and Vendor Certificate | Externally | Plaintext | Entry: Hard Coded in ASIC ROM Output: N/A |
| SWAK | HSM SW Download Key | ECDSA used to validate firmware | Externally | Plaintext | Entry: Hard coded in SDU Output: N/A |

*Figure 7 – Public Key Table*

The following table describes the modes of access for each key to each role supported by the module. The modes of access are defined as:

| Roles | | Services | CSP Modes of Access |
|---|---|---|---|
| CO | TU | | |
| X | X | Split Key | Splits AK and PK |
| X | X | Join Key | Joins AK and PK |
| X | X | Generate | Generates AK and PK |
| X | X | Export Key | Exports AK and PK |
| X | X | Load Key | Loads AK and PK |
| X | X | Delete Key | Used to remove AK and PK |
| X | X | Hash | N/A |
| X | X | Modular Exponentiation | N/A |
| X | X | Load Parameters | N/A |
| X | X | Reinit | Zeroizes Secret and Private key data |
| X | X | Setup Download Data | Uses SWAK for authentication |
| X | X | Setup Download Chunk | Uses SWAK for authentication |
| X | X | Download Chunk | N/A |
| X | X | Get Random Number | N/A |
| X | X | Encrypt | Uses PK |
| X | X | Decrypt | Uses PK |
| X | X | Firmware Update | Uses SWAK |
| X | X | Decrypt Compare | Uses PK |
| X | X | Decrypt Encrypt | Uses PK |
| X | X | Sign | Uses AK |
| X | X | Verify | Uses AK |
| X | X | Derive Key | Uses AK and PK |
| X | X | Get Key List | N/A |
| X | X | Get Parameters | N/A |
| X | X | Get Status | N/A |
| X | X | Get HW Status | N/A |
| X | X | Get Versions | N/A |
| X | X | GetTimeDrift | N/A |
| X | X | SetTimeDrift | N/A |

| Roles | | Services | CSP Modes of Access |
|---|---|---|---|
| CO | PU | | |
| X | X | Reboot | N/A |
| X | X | Set Counter | N/A |
| X | X | Update Counters | N/A |
| X | X | Get Counter Record | Uses AK |
| X | X | Perform Diagnostic Test | N/A |
| X | X | Read Log | N/A |
| X | X | Add Log Entry | N/A |
| X | X | Delete Log | N/A |
| X | X | Read Mfg Data | N/A |
| X | X | Clear Private Information | N/A |

*Figure 8 – CSP Modes of Access*

# 9. Operational Environment

The FIPS 140-2 Area 6 Operational Environment requirements for the module are not applicable because the device does not contain a modifiable operational environment.

# 10. Security Rules

This section documents the security rules enforced by the module to implement the security requirements of this FIPS 140-2 Level 3 module.

- The module shall not process more than one request at a time (i.e., single threaded). While processing a transaction, prior to returning a response, the module will ignore all other inputs to the module. No output is performed until the transaction is completed, and the only output is the transaction response.
- The module shall authenticate identities based on knowledge of a shared secret.
- All keys generated in the module shall have at least 80-bits of strength.
- All methods of key generation shall be at least as strong as the key being generated.
- All methods of key establishment shall be at least as strong as the key being established.
- The module shall not provide a bypass state where plaintext information is just passed through the module.
- The module shall not support a maintenance mode.
- The module shall not output any secret or private key in plaintext form.

- The module shall not accept any secret or private key in plaintext form outside of manufacturing.
- There shall be no manual entry of keys into the system.
- There shall be no entry or output of split keys from the system except in KRA.
- There shall be no key archiving.
- Keys shall be either generated via an Approved method or entered into the system through FIPS Approved processes.
- Once a module has been zeroized, it must be returned to the factory for software loading and parameterizing prior to being usable by a customer.
- The module shall support the following conditional tests:
    - Pairwise consistency test for DSA key pair generation
    - Pairwise consistency test for ECDSA key pair generation
    - Continuous RNG test for the DBRG – Stuck Seed, Stuck Number
    - ECDSA Signature Verification - Firmware Load Test
    - ECDSA Public Key Validation as part of SP 800-56A Key Agreement Protocol
- The module shall support power up self-tests, which can also be run as requested by the user, include:
    - Firmware Integrity Tests:
        - Digital Signature - ECDSA 256
    - Sigma ASIC Power On Self-Tests (POST)
        - TDES Known Answer Test
        - DSA Verification Known Answer Test
        - ECDSA Verification Known Answer Test
        - SHA-1 Known Answer Test
        - SHA-256 Known Answer Test
        - AES Engine Known Answer Test (128, 256)
        - RSA Sign/Verify Known Answer Test
    - Critical functions tests:
        - RTC Test
        - Sigma ASIC POST
        - Bram Pattern Test
    - Cryptographic Algorithm Known Answer Tests: (DAL POST)
        - DSA Pairwise Consistency Test
        - ECDSA Pairwise consistency
        - AES Key Wrap / Unwrap Known Answer Test
        - AES Encrypt / Decrypt Known Answer Test

        - HMAC SHA-1 Known Answer Test
        - HMAC SHA-256 Known Answer Test
        - KAS SP800-56A (C(2, 0, ECC CDH)) Known Answer Test
        - HASH DRBG SP800-90 Known Answer Test
- Self-tests may be initiated by the following means:
    - Perform Diagnostic Test service
    - Physically recycling the module's power
- The status of self-tests shall be available via the Get HW Status service.

# 11. Physical Security Policy

The Cygnus XHSM ASIC is a single chip cryptographic module. The module is covered by a hard opaque encapsulant material. Attempts to penetrate the ASIC device packaging have a high probability of causing serious damage to the module.

Hardness testing was performed at ambient temperature, at 65°C, and at 5°C.

The module protects key material from unauthorized disclosure.

# 12. Mitigation of Other Attacks Policy

The module has not been designed to mitigate any specific attacks outside the scope of FIPS 140-2.

# 13. References

The following documents are referenced by this document, are related to it, or provide background material related to it:

- Digital Signature Standard (DSA) – FIPS PUB 186-2, January 27, 2000, including change notice of October 5, 2001
- Digital Signature Standard (DSA) – FIPS PUB 186-3, November 2008
- Advanced Encryption Standard (AES) FIPS PUB 197, November 26, 2001
- Recommendation for Block Cipher Modes of Operation, Methods and Techniques, Special Publication 800-38A, December 2001.
- Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication 800-67, May 2004.
- The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standards Publication 198, March 06, 2002
- Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised), Special Publication 800-90, March 2007.

- AES Key Wrap Specification, November 2001
- Secure Hash Standard – FIPS PUB 180-3, October 2008
- NIST SP 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography – March 2007
- Security Requirements for Cryptographic Modules – FIPS PUB 140-2, Change Notices December 3, 2002

# 14. Acronyms

| | |
|---|---|
| AES | Advanced Encryption Standard |
| DAL | Device Abstraction Layer |
| DH | Diffie-Hellman |
| DSA | Digital Signature Algorithm |
| ECC CDH | Elliptic Curve Cryptography Cofactor Diffie-Hellman |
| ECDH | Elliptic Curve Diffie-Hellman |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| HMAC | Hashed Message Authentication Code |
| HSM | Hardware Security Module |
| KAS | Key Agreement Scheme |
| KRA | Key Root Authority |
| RSA | Rivest Shamir Adleman |
| RTC | Real Time Clock |
| SDU | Software Download Utility |
| SHA | Secure Hash Algorithm |
| TDES | Triple-DES |

# Revision History

| Version | Date | Revision Description |
|---|---|---|
| 0.1 | 10/15/2010 | Original Document |
| 1.00 | 6/15/2011 | Updates from internal review |
| 1.01 | 8/15/2011 | |
| 1.02 | 9/30/2011 | Update based on Infogard comments.<br>Update to match current messaging. |
| 1.03 | 3/13/2011 | Update based on Infogard comments |
| 1.04 | 3/20/12 | Consolidated lists of self test in section 10 |
| 1.05 | 3/21/12 | Incorporated comments from Infogard review and Operational Test.<br>Combined lists of approved algorithms in section 3 into one table.<br>Removed "DecryptCompare" command, which is not supported by the DAL.<br>Section 6: Added SetTimeDrift command to Unauthenticated Services |
| 1.06 | 4/26/12 | Section 7: Change HSM Administrator to Crypto Officer and clarified SW download key descriptions.<br>Section 8: Added Session Key (SK) to figure 6 and removed SMAK from figure 8. |
| 1.07 | 5/8/12 | Incorporated feedback from InfoGard review. |
| 1.08 | 12/11/12 | Changes in response to CMVP comments<br>Section 1: Added "P/N" to the first line of the table.<br>Section 11: Added statement with temperatures for hardness testing. |
| | | |
| | | |
| | | |
| | | |
| | | |