nCipher

www.ncipher.com

nShield F2 500+ & nShield F2 1500+ & nShield F2 6000+

Non-proprietary Security Policy for FIPS 140-2 Level 2

Version: 1.0

Date: 1st October 2020

Contents

1 Introduction

1.1 Scope

This document defines the non-proprietary Security Policy enforced by the nShield Hardware Security Module, i.e. the Cryptographic Module, to meet with the security requirements in FIPS 140-2.

The following product hardware variants and firmware version(s) are in scope of this Security Policy.

Variant name	Marketing model number	Firmware version
nShield F2 500+	nC3423E-500[1]	12.50.8
nShield F2 1500+	nC3423E-1K5[1]	
nShield F2 6000+	nC3423E-6K0[1]	

[1] These modules are labelled with the model number A-025001-L

Variants

All modules are supplied at build standard “N”

1.2 Security level

The Cryptographic Module meets overall **FIPS 140-2 Security Level 2**. The following table specifies the security level in detail.

Table 1 Security level of security requirements

Security requirements section	Level
Cryptographic Module Specification	2
Module Ports and Interfaces	2
Roles, Services and Authentication	3
Finite State Model	2

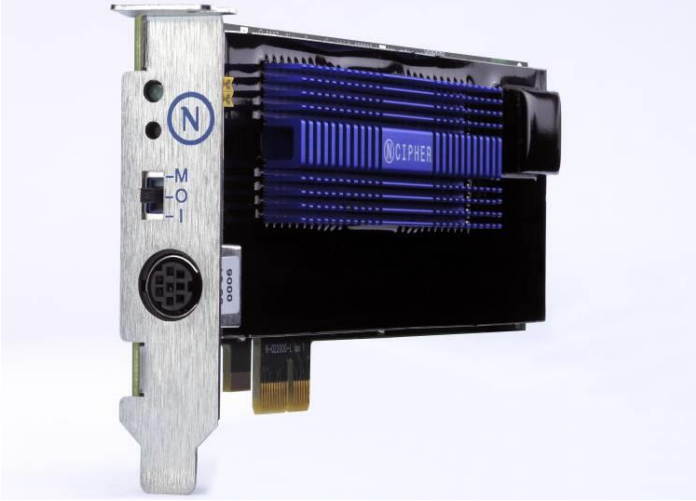
Physical Security	3
Operational Environment	N/A
Cryptographic Key Management	2
EMI/EMC	3
Self-Tests	2
Design Assurance	3
Mitigation of Other Attacks	N/A

1.3 Cryptographic module description

The nShield Hardware Security Module (HSM) is a multi-chip embedded Cryptographic Module as defined in FIPS 140-2, which comes in a PCI express board form factor protected by a tamper resistant enclosure, and performs encryption, digital signing, and key management on behalf of an extensive range of commercial and custom-built applications including public key infrastructures (PKIs), identity management systems, application-level encryption and tokenization, SSL/TLS, and code signing.

The Figure below shows the nShield Solo+ HSM.

Figure 1 nShield Solo+



The cryptographic boundary is delimited in red in the images in the table below. It is delimited by the heat sink and the outer edge of the potting material on the top and bottom of the PCB.

The Cryptographic Module provides the following physical ports and interfaces, which remain outside of the cryptographic boundary:

- PCIe bus (data input/output, control input, status output and power). The services provided by the module are transported through this interface.
- Status LED (status output)
- Mode switch (control input)
- Clear button (control input)
- PS/2 serial connector for connecting a smartcard reader (data input/output).
- 14-way header (data input/output, control input, status output) which provides alternative connections for the mode switch, clear button, status LED and serial connector.
- Dual configuration switches (control input), are a set of two jumpers which enable the mode switch and enable the remote mode switching.
- Battery (power), providing power backup.
- Heat fan control signal.

The PCB traces coming from those connectors transport the signals into the module's cryptographic boundary and cannot be used to compromise the security of the module.

The top cover, heat fan and the battery are outside the module's cryptographic boundary and cannot be used to compromise the security of the module.

Table 2 Cryptographic module boundary

1.4 Operational environment

The FIPS 140-2 Operational Environment requirements are not applicable because the cryptographic module contains a limited operational environment.

2 Cryptographic Functionality

2.1 Security World overview

The security model of the module is based around the Security World concept for secure management of cryptographic keys.

A Security World includes:

- An Administrator Card Set (ACS), a set of Administrator smart cards used to perform administrative operations,
- Optionally, one or more Operator Card Sets (OCSs), a set or sets of Operator smart cards used to control access to application keys and to authorise certain operations,
- Optionally, a set of Softcards used to control access to application keys,
- Key Blobs, which contain cryptographic keys and their associated Access Control List (ACL), whose confidentiality and integrity are protected by approved algorithms. They are stored outside the Cryptographic Module.

2.2 Keys and Critical Security Parameters

The Cryptographic Module uses and protects the following keys and Critical Security Parameters (CSPs):

Table 3 CSP table

CSP	Type	Description	Generation	Input	Output	Storage	Zeroization
KRE - Recovery Confidentiality Key	RSA 3072-bit	Key used to protect recovery keys (KR). KTS (vendor affirmed)	DRBG	Load Blob - encrypted with LT	Make Blob - encrypted with LT	Ephemeral, stored in volatile RAM.	Initialize Unit
KR - Recovery Key	AES 256-bit	Key used to derive (using SP 800-108 KDF in counter mode) the keys Ke (AES 256-bit) and Km (HMAC-SHA256) that protect an archive copy of an application key. • AES cert#C754	DRBG	Load Blob - encrypted with KRE	Make Blob - encrypted with KRE	Ephemeral, stored in volatile RAM.	Initialize Unit, Clear Unit, power cycle or reboot.
Impath session keys	AES 256-bit in CBC mode. Integrity with HMAC SHA-256.	Used for secure channel between two modules. It consists of a set of four session keys used in an Impath session for encryption, decryption, MAC generation and MAC validation. • AES cert#C754 • HMAC cert#C754	3072-bit DH key exchange with one-step KDF with SHA-1 between two modules.	No	No	Ephemeral, stored in volatile RAM.	Clear Unit, new session, power cycle or reboot.

KJSO - JSO key	DSA 3072-bit	nShield Junior Security Officer key used with its associated certificate to perform the operations allowed by the NSO. • DSA cert#C754	DRBG	Load Blob - encrypted with LT	Make Blob - encrypted with LT	Ephemeral, stored in volatile RAM.	Destroy, Initialize Unit, Clear Unit, power cycle or reboot.
KA - Application key	AES 128, 192, 256 bits TDES 192 bits HMAC with key sizes >= 112 bits RSA with key sizes >= 2048 bits DSA, DH with key sizes >= 2048 bits ECDSA, ECDH, EC MQV with curves: • P-224, P-256, P-384, P-521 • K-233, K-283, K-409, K-571 • B-233, B-283, B-409, B-571	Keys associated with a user to perform cryptographic operations, that can be used with one of the following validated algorithms: • AES and KTS cert#C754 • HMAC cert#C754 • RSA cert #C754 • DSA cert #C754 • ECDSA cert #C754 • Key Agreement (KAS) cert#C754 • KBKDF cert#C754 • KTS (vendor affirmed)	DRBG	Load Blob - encrypted with LT or KR	Make Blob - encrypted with LT or KR	Ephemeral, stored in volatile RAM.	Destroy, Initialize Unit, Clear Unit, power cycle or reboot

KM - Module Key	AES 256-bit	Key used to protect logical tokens and associated module Key Blobs. • AES cert#C754	DRBG	Load Blob - encrypted with LT	Make Blob - encrypted with LT	Non-volatile memory	Initialize Unit
KML - Module Signing Key	DSA 3072-bit	Module Signing Key used by the module to sign key generation and module state certificates. When the nShield module is initialized, it automatically generates this key that it uses to sign certificates using DSA with SHA-256. This key is only ever used to verify that a certificate was generated by a specific module. • DSA cert #C754	DRBG	No	No	Non-volatile memory	Initialize Unit
KNSO - NSO key	DSA 3072-bit	nShield Security Officer key used for NSO authorisation and Security World integrity. Used to sign Delegation Certificates and to directly authorize commands during recovery operations • DSA cert #C754	DRBG	Load Blob - encrypted with LT	Make Blob - encrypted with LT	Ephemeral, stored in volatile RAM.	Destroy, Initialize Unit, Clear Unit, power cycle or reboot.
LT - Logical Token	AES 256-bit	Key used to derive the keys that are used to protect token protected key blobs. Logical Tokens are split in shares (encrypted with Share Key) between one or more smartcards or a softcard, using the Shamir Secret Sharing scheme. • AES cert #C754 • KDF cert #C754	DRBG	Read Share - encrypted with Share Key	Write Share - encrypted with Share Key	Ephemeral, stored in volatile RAM.	Destroy, Initialize Unit, power cycle or reboot

		• HMAC cert #C754					
Share Key	AES 256-bit	Protects a share when written to a smartcard or softcard. This key is used to derive (using SP 800-108 AES CTR KDF) the keys Ke (AES 256-bit) and Km (HMAC-SHA256) that wrap the share. • AES cert #C754 • KDF cert #C754 • HMAC cert #C754	DRBG	No	No	Ephemeral, stored in volatile RAM.	N/A
Remote Administration session keys	AES 256-bit in CBC mode Integrity with CMAC	Used for secure channel between the module and a smartcard. This is a set of four AES 256-bit session keys, namely K_{m-e} (for encrypting data send to the smartcard), K_{c-e} (for decrypting data from the smartcard), K_{m-a} (for CMAC generation) and K_{c-a} (for CMAC verification). • AES cert #C754	ECDH P-521 key agreement with SP 800-108 KDF in counter mode.	No	No	Ephemeral, stored in volatile RAM.	Clear Unit, new session, power cycle or reboot.
KAL - Key Audit Logging	DSA 3072-bit	Used for signing the log trail. • DSA cert #C754	DRBG	No	No	Non-volatile memory	Initialize Unit
DRBG internal state	Hash_DRBG	The module uses the Hash_DRBG with SHA-256 compliant with SP800-90A. • Hash DRBG cert #C754	Entropy source	No	No	Ephemeral, stored in volatile RAM.	Clear Unit, power cycle or reboot.
DRBG entropy input	776 bits	Entropy input string used to initialize and re-seed the DRBG.	Entropy source	No	No	Ephemeral, stored in volatile RAM.	Clear Unit, power cycle or reboot.

The following table describes the public keys handled by the module:

Table 4 Public key table

Public Key	Type	Description	Generation	Input	Output	Storage
Firmware Integrity key (KFI)	DSA 3072-bit	Public key used to ensure the integrity of firmware updates. The module validates the signature before new firmware is written to non-volatile storage. • DSA cert #C754	At nCipher	Firmware update	No	In firmware
KJWAR	ECDSA P-521	nCipher root warranting public key for Remote Administrator Cards and Remote Operator Cards • ECDSA cert #C754	At nCipher	Firmware update	None	Persistent storage in plaintext inside the module (EEPROM)
Application keys public key	See description	Public keys associated with private Application keys: • RSA cert #C754 • DSA cert #C754 • ECDSA cert #C754 • Key Agreement (KAS) cert#C754 • KTS (vendor affirmed)	At creation of the application key	Load Blob - encrypted with LT	Key export	Stored in the key blob of the application key
KJSO public key	DSA 3072-bit	Public key associated to KJSO • DSA cert #C754	At creation of the KJSO	Load Blob - encrypted with LT	Key export	Public key hash stored in the module persistent storage

KNSO public key	DSA 3072-bit	Public key associated to KNSO • DSA cert #C754	At creation of the KNSO	Load Blob - encrypted with LT	Key export	Public key hash stored in the module persistent storage
KML public key	DSA 3072-bit	Public key associated to KML • DSA cert #C754	At creation of KML	No	Key export	Public key hash stored in the module persistent storage
KAL public key	DSA 3072-bit	Public key associated to KAL • DSA cert #C754	At creation of KAL	No	Included in the audit trail	Public key hash stored in the module persistent storage
KRE public key	RSA 3072-bit	Public key associated to KRE • KTS (vendor affirmed)	At creation of the KNSO	Load Blob - encrypted with LT	Key export	Stored in a key blob
FET public key	DSA 1024-bit	Feature Enable Tool (FET) public key used to verify FET certificates • DSA cert #C754	At nCipher	Firmware update	No	Persistent storage in plaintext inside the module (EEPROM)
Impath DH public key	DH 3072-bit	Public key from peer used in the Impath DH key agreement • KAS-FFC cert #C754	No	Loaded with Cmd_ImpathKXFinish	No	Ephemeral, stored in volatile RAM.
Remote Administration ECDH public key	NIST P-521	Public key from peer used in the Remote Administration ECDH key agreement • KAS-ECC cert #C754	No	Loaded with Cmd_DynamicSlotExchangeAPDUs	No	Ephemeral, stored in volatile RAM.

2.3 Supported cryptographic algorithms

2.3.1 FIPS Approved or Allowed Algorithms

The following tables describe the Approved or allowed cryptographic algorithms supported by the Cryptographic Module.

Table 5 Approved algorithms

Cert #	Algorithm	Standard	Details
#C754	AES	FIPS 197 SP800-38A SP800-38D SP800-38B	ECB (e/d; 128 , 192 , 256); CBC (e/d; 128 , 192 , 256); CTR (int only; 256) CMAC (Generation/Verification) (KS: 128; Block Size(s): ; Msg Len(s) Min: 0 Max:2^16 ; Tag Len(s) Min: 16 Max: 16) (KS: 192; Block Size(s): ; Msg Len(s) Min: 0 Max: 2^16 ; Tag Len(s) Min: 16 Max: 16) (KS: 256; Block Size(s): ; Msg Len(s) Min: 0 Max: 2^16 ;Tag Len(s) Min: 16 Max: 16) GCM (KS: AES_128 (e/d) Tag Length(s): 128 120 112 104 96 64 32) (KS:AES_192 (e/d) Tag Length(s): 128 120 112 104 96 64 32) (KS: AES_256 (e/d) Tag Length(s): 128 120 112 104 96 64 32) IV Generated: (Internal (using Section 8.2.2)) ; PT Lengths Tested: (0 , 1024 ,1024) ; AAD Lengths tested: (1024 , 1024) ; 96BitIV_Supported ; OtherIVLen_Supported DRBG: Val#C754
#C754	KTS	SP800-38D SP800-38F	GCM (KS: AES_128 (e/d) Tag Length(s): 128 120 112 104 96 64 32) (KS:AES_192 (e/d) Tag Length(s): 128 120 112 104 96 64 32) (KS: AES_256 (e/d) Tag Length(s): 128 120 112 104 96 64 32) IV Generated: (Internal (using Section 8.2.2)) ; PT Lengths Tested: (0 , 1024 ,1024) ; AAD Lengths tested: (1024 , 1024) ; 96BitIV_Supported ; OtherIVLen_Supported DRBG: Val#C754 KW (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 4096)
#C754	Triple-DES *Note: The user is responsible to comply with the maximum use of the same key for encryption encryption operations, limited to 2^20 or 2^16, as defined in Implementation*	SP800-67	TECB(KO 1 e/d,) ; TCBC(KO 1 e/d,)

	Guidance A.13 SP 800-67rev1 Transition.		
#C754	SHA	FIPS 180-4	SHA-1 (BYTE-only) SHA-224 (BYTE-only) SHA-256 (BYTE-only) SHA-384 (BYTE-only) SHA-512 (BYTE-only) Implementation does not support zero-length (null) messages.
#C754	HMAC with SHA	FIPS 198-1	HMAC-SHA1 (Key Sizes Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#C754 HMAC-SHA224 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#C754 HMAC-SHA256 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#C754 HMAC-SHA384 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#C754 HMAC-SHA512 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHSVal#C754
#C754	RSA	FIPS 186-4	FIPS186-4: 186-4KEY(gen): FIPS186-4_Random_e PGM(ProbRandom: (2048 , 3072 , 4096) PPTT:(C.3) ALG[RSASSA-PKCS1_V1_5] SIG(gen) (2048 SHA(224 , 256 , 384 , 512)) (3072 SHA(224 , 256 , 384 , 512)) (4096 SHA(224 , 256 , 384 , 512) SIG(Ver) (1024 SHA(1 , 224 , 256 , 384 , 512)) (2048 SHA(1 , 224 , 256 , 384 , 512)) (3072 SHA(1 , 224 , 256 , 384 , 512)) (4096 SHA(1 , 224 , 256 , 384 , 512)) [RSASSA-PSS]: Sig(Gen): (2048 SHA(224 SaltLen(28) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(224 SaltLen(28) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64) 4096 SH(224 SaltLen(28) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) Sig(Ver): (1024 SHA(1 SaltLen(20) , 224 SaltLen(28) , 256 SaltLen(32) , 384 SaltLen(48))) (2048 SHA(1 SaltLen(20) , 224 SaltLen(28) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(1 SaltLen(20) , 224 SaltLen(28) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (4096 SHA(1 SaltLen(20) , 224 SaltLen(28) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) SHA Val#C754 DRBG: Val#C754
Vendor affirmed	KTS	SP 800-56B	KTS-OAEP-basic with SHA-224, SHA-256, SHA-384, SHA-512 (key establishment methodology provides between 112 and 256 bits of encryption strength)

#C754	DSA	FIPS 186-4	FIPS186-4: PQG(gen)PARMS TESTED: [(2048, 224)SHA(224); (2048,256)SHA(256); (3072,256) SHA(256)] PQG(ver)PARMS TESTED: [(1024,160) SHA(1); (2048,224) SHA(224); (2048,256) SHA(256); (3072,256) SHA(256)] KeyPairGen: [(2048,224) ; (2048,256) ; (3072,256)] SIG(gen)PARMS TESTED: [(2048,224) SHA(224 , 256 , 384 , 512); (2048,256) SHA(256 , 384 , 512); (3072,256) SHA(256 , 384 , 512);] SIG(ver)PARMS TESTED: [(1024,160) SHA(1 , 224 , 256 , 384 , 512); (2048,224) SHA(224 , 256 , 384 , 512); (2048,256) SHA(256 , 384 , 512); (3072,256) SHA(256 , 384 , 512)] SHS: Val#C754 DRBG: Val#C754
#C754	ECDSA	FIPS 186-4	FIPS186-4: PKG: CURVES(P-224 P-256 P-384 P-521 K-233 K-283 K-409 K-571 B-233 B-283 B-409 B-571 ExtraRandomBits) PKV: CURVES(ALL-P ALL-K ALL-B) SigGen: CURVES(P-224: (SHA-224, 256, 384, 512) P-256: (SHA-256, 384, 512) P-384: (SHA-384, 512) P-521: (SHA-512) K-233: (SHA-224, 256, 384, 512) K-283: (SHA-256, 384, 512) K-409: (SHA-384, 512) K-571: (SHA-512) B-233: (SHA-224, 256, 384, 512) B-283: (SHA-256, 384, 512) B-409: (SHA-384, 512) B-571: (SHA-512)) SigVer: CURVES(P-192: (SHA-1, 224, 256, 384, 512) P-224: (SHA-224, 256, 384, 512) P-256: (SHA-256, 384, 512) P-384: (SHA-384, 512) P-521: (SHA-512) K-163: (SHA-1, 224, 256, 384, 512) K-233: (SHA-224, 256, 384, 512) K-283: (SHA-256, 384, 512) K-409: (SHA-384, 512) K-571: (SHA-512) B-163: (SHA-1, 224, 256, 384, 512) B-233: (SHA-224, 256, 384, 512) B-283: (SHA-256, 384, 512) B-409: (SHA-384, 512) B-571: (SHA-512)) SHS: Val#C754 DRBG: Val#C754
#C754	Key Agreement Component	SP800-56A	KAS-FFC-Component: (FUNCTIONS INCLUDED IN IMPLEMENTATION: KPG Partial Validation) SCHEMES: Ephem: (KARole: Initiator / Responder) FB FC OneFlow: (KARole: Initiator / Responder) FB FC Static: (KARole: Initiator / Responder) FB FC DSA Val#C754, SHS Val#C754, DRBG Val#C754 KAS-ECC-Component: (FUNCTIONS INCLUDED IN IMPLEMENTATION: KPG Partial Validation) SCHEMES: FullMQV: (KARole: Initiator / Responder) EB: P-224 EC: P-256 ED: P-384 EE: P-521 EphemUnified: (KARole: Initiator / Responder) EB: P-224 EC: P-256 ED: P-384 EE: P-521 OnePassDH: (KARole: Initiator) EB: P-224 EC: P-256 ED: P-384 EE: P-521 StaticUnified: (KARole: Initiator / Responder) EB: P-224 EC: P-256 ED: P-384 EE: P-521 ECDSA Val#C754, SHS Val#C754, DRBG Val#C754

#C754	KBKDF	SP800-108	CTR_Mode: (Llength(Min16 Max16) MACSupported([CMACAES256]) LocationCounter([BeforeFixedData]) rlength([8])) AES Val#C754 DRBG Val#C754
#C754	DRBG	SP800-90A	Hash_Based DRBG: [Prediction Resistance Tested: Not Enabled (SHA-256) (SHS Val#C754)]
Vendor affirmed	CKG	SP800-133	Symmetric keys are generated using the unmodified output of the approved DRBG.

Table 6 Allowed algorithms

Algorithm
Diffie-Hellman (CVL Cert. #C754, key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)
EC Diffie-Hellman (CVL Cert. #C754, key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)
EC MQV (CVL Cert. #C754, key agreement; key establishment methodology provides between 112 and 256 bits of encryption strength)
Allowed Non-deterministic Random Number Generator (NDRNG). NDRNG is used to seed the approved DRBG. The module generates a minimum of 256 bits of entropy for key generation.

2.3.2 Non-Approved Algorithms

The following table describes the non-approved cryptographic algorithms supported by the Cryptographic Module in non-Approved mode.

Table 7 Non-approved algorithms

Algorithm
Symmetric encryption and decryption
DES
Two-key Triple DES encryption, MAC generation
AES GCM with externally generated IV

AES CBC MAC
Aria
Camellia
Arc Four (compatible with RC4)
CAST 256 (RFC2612)
SEED (Korean Data Encryption Standard)
Asymmetric
Raw RSA data encryption and decryption
KTS-OAEP-basic with SHA-256 with key size less than 2048 bits
ElGamal (encryption using Diffie-Hellman keys)
KCDSA (Korean Certificate-based Digital Signature Algorithm)
RSA digital signature generation with SHA-1 or key size less than 2048 bits
DSA digital signature generation with SHA-1 or key size less than 2048 bits
ECDSA digital signature generation with SHA-1 or curves P-192, K-163 , B-163, Brainpool
DH with key size p < 2048 bits or q < 224 bits
ECDH with curves P-192, K-163, B-163, Brainpool
EC MQV with curves P-192, K-163 or B-163
Deterministic DSA compliant with RFC6979
Ed25519 public-key signature
X25519 key exchange
Hash
HAS-160

MD5
RIPEMD-160
Tiger
Message Authentication Codes
HMAC with MD5, RIPEMD-160 and Tiger
HMAC with key size less than 112 bits
Other
TLS 1.0 and SSL 3.0 KDF (The protocols SSL, TLS shall not be used when operated in the Approved mode. In particular, none of the keys derived using this key derivation function can be used in the Approved mode).
PKCS#8 padding
EMV support: Cryptogram (ARQC) generation and verification (includes EMV2000, M/Chip 4 and Visa Cryptogram Version 14, EMV 2004, M/Chip 2.1, Visa Cryptogram Version 10) Watchword generation and verification
Hyperledger client side KDF

3 Roles and Services

3.1 Roles

The Cryptographic Module supports the following roles:

- nShield Security Officer (NSO)
- Junior Security Officer (JSO)
- User

nShield Security Officer (NSO)

This role is represented by Administrator Card holders, which have access to KNSO and are responsible for the overall management of the Cryptographic Module.

To assume this role, an operator or group of operators need to present a quorum *m of N* of smartcards, and the KNSO Key Blob. Each operator is identified by its individual smartcard, which contains a unique logical token share.

Junior Security Officer (JSO)

This role is represented by either Administrator Card or Operator Card holders with a KJSO and an associated Delegation Certificate signed by KNSO, authorising a set of commands.

To assume this role, an operator or group of operators need to present a quorum *m of N* of smartcards and the associated Delegation Certificate. Each operator is identified by its individual smartcard or Softcard, which contains a unique logical token share.

User

This role is represented by Application key owners, which are authorised to perform approved services in the module using those keys.

To assume this role, an operator or group of operators need to present a quorum *m of N* of smartcards or a Softcard, and the Key Blob. Each operator is identified by its individual Smartcard or Softcard, which contains a unique logical token share.

3.2 Strength of authentication mechanisms

Table 8 Strength of authentication table

Authentication mechanism	Type of authentication	Strength of Mechanism
Smartcard	Identity based	A logical token share stored in a Smartcard or Softcard is encrypted and MAC'ed. An attacker would need to guess the encrypted share value and the associated MAC in order to be able to load a valid Logical token share into the module. This requires, as a minimum, guessing a 256-bit HMAC-SHA256 value, which gives a probability of 2^-256. This probability is less than 10^-6. The module can process around 2^16 commands per minute. This gives a probability of success in a one minute period of 2^-240, which is less than 10^-5.
Softcard	Identity based	

3.3 Services

The following table describes the services provided by the Cryptographic Module and the access policy.

The Access column presents the access level given to the CSP, R for Read, W for Write, Z for Zeroise

Table 9 Service table

Service	Description	Authorized roles	Access	CSPs
Big number operation Cmd_BignumOp	Performs an operation on a large integer.	Unauthenticated	-	None
Make Blob Cmd_MakeBlob	Creates a Key blob containing the key. Note that the key ACL needs to authorize the operation.	User / JSO / NSO	W	KA, KRE, KR, KJSO, KM, KNSO, LT
Buffer operations Cmd_CreateBuffer Cmd_LoadBuffer	Mechanism for loading of data into the module volatile memory. The data can be loaded in encrypted form which can be decrypted inside the module with a key that has been previously loaded.	Unauthenticated	R	KA
Bulk channel Cmd_ChannelOpen Cmd_ChannelUpdate	Provides a bulk processing channel for encryption / decryption, MAC generation / verification and signature generation / verification.	User	R	KA

Check User Action Cmd_CheckUserAction	Determines whether the ACL associated with a key allows a specific operator defined action.	User / JSO / NSO	R	KNSO, KJSO; KA
Clear Unit Cmd_ClearUnit	Zeroises all keys, tokens and shares that are loaded into the module. Will cause the module to reboot and perform self-tests.	Unauthenticated	Z	KA, KR, Impath keys, KJSO, remote administration session keys
Set Module Key Cmd_SetKM	Allows a key to be stored internally as a Module key (KM) value. The ACL needs to authorize this operation.	NSO	W	KM
Remove Module Key Cmd_RemoveKM	Deletes a given KM from non-volatile memory.	NSO	Z	KM
Duplicate key handle Cmd_Duplicate	Creates a second instance of a Key with the same ACL and returns a handle to the new instance. Note that the source key ACL needs to authorize this operation.	User / JSO / NSO	R	KA
Enable feature Cmd_StaticFeatureEnable	Enables the service. This service requires a certificate signed by the Master Feature Enable key.	Unauthenticated	-	None
Encryption / decryption Cmd_Encrypt Cmd_Decrypt	Encryption and decryption using the provided key handle.	User	R	KA
Erase from smartcard /softcard Cmd_EraseFile Cmd_EraseShare	Removes a file or a share from a smartcard or softcard	NSO / JSO / User	-	None
Format Token Cmd_FormatToken	Formats a smartcard or a softcard.	Unauthenticated	-	None
File operations Cmd_FileCopy Cmd_FileCreate Cmd_FileErase	Performs file operations in the module.	NSO / JSO	-	None

Cmd_FileOp				
Firmware Authenticate Cmd_FirmwareAuthenticate	Reports firmware version, using a zero knowledge challenge response protocol based on HMAC. The protocol generates a random value to use as the HMAC key.	Unauthenticated	-	None
Force module to fail Cmd_Fail	Causes the module to enter a failure state.	Unauthenticated	-	None
Foreign Token open Cmd_ForeignTokenOpen	Opens a channel for direct data access to a Smartcard Requires Feature Enabled.	NSO / JSO	-	None
Foreign Token command Cmd_ForeignTokenCommand	Sends an ISO-7816 command to a smartcard over the channel opened by ForeignTokenOpen.	Unauthenticated	-	None
Firmware Update Cmd_Maintenance Cmd_ProgrammingBegin Cmd_ProgrammingBeginChunk Cmd_ProgrammingLoadBlock Cmd_ProgrammingEndChunk Cmd_ProgrammingEnd Cmd_ProgrammingGetKeyList	Perform a firmware update. Restricted service to nCipher signed Firmware.	Unauthenticated	R	KFI
Generate prime number Cmd_GeneratePrime	Generates a random prime number.	Unauthenticated	R, W	DRBG internal state
Generate random number Cmd_GenerateRandom	Generates a random number from the Approved DRBG.	Unauthenticated	R, W	DRBG internal state
Get ACL Cmd_GetACL	Get the ACL of a given key.	User	R	KA
Get key application data Cmd_GetAppData	Get the application data field from a key.	User	R	KA

Get challenge Cmd_GetChallenge	Get a random challenge that can be used in fresh certificates.	Unauthenticated	R, W	DRBG internal state
Get KLF2 Cmd_GetKLF2	Get a handle to the Module Long Term (KLF2) public key.	Unauthenticated	-	None
Get Key Information Cmd_GetKeyInfo Cmd_GetKeyInfoEx	Get the type, length and hash of a key.	NSO / JSO / User	R	KA
Get module signing key Cmd_GetKML	Get a handle to the KML public key.	Unauthenticated	R	KML
Get list of slot in the module Cmd_GetSlotList	Get the list of slots that are available from the module.	Unauthenticated	-	None
Get Logical Token Info Cmd_GetLogicalTokenInfo Cmd_GetLogicalTokenInfoEx	Get information about a Logical Token: hash, state and number of shares.	NSO / JSO / User	R	LT
Get list of module keys Cmd_GetKMList	Get the list of the hashes of all module keys and the KNSO.	Unauthenticated	R	KM, KNSO
Get module state Cmd_GetModuleState	Returns unsigned data about the current state of the module.	Unauthenticated	-	None
Get real time clock Cmd_GetRTC	Get the current time from the module Real Time Clock.	Unauthenticated	-	None
Get share access control list Cmd_GetShareACL	Get the Share's ACL.	NSO / JSO / User	R	Share Key
Get Slot Information Cmd_GetSlotInfo	Get information about shares and files on a Smartcard that has been inserted in a module slot.	Unauthenticated	-	None
Get Ticket Cmd_GetTicket	Get a ticket (an invariant identifier) for a key. This can be passed to another client or to a SEE World which can redeem it	NSO / JSO / User	-	None

	using Redeem Ticket to obtain a new handle to the object.			
Initialize Unit Cmd_InitializeUnit Cmd_InitializeUnitEx	Causes a module in the pre-initialization state to enter the initialization state. When the module enters the initialization state, it erases all Module keys (KM), the module's signing key (KML), and the hash of the Security Officer's keys, HKNSO. It then generates a new KML and KM.	Unauthenticated	Z	KA, KRE, KR, KJSO, KM, KAL, KML, KNSO, LT
Insert a Softcard Cmd_InsertSoftToken	Allocates memory on the module that is used to store the logical token share and other data objects.	Unauthenticated	R	Share Key
Remove a Softcard Cmd_RemoveSoftToken	Removes a Softcard from the module. It returns the updated shares and deletes them from the module's memory.	Unauthenticated	Z	Share Key
Impath secure channel Cmd_ImpathGetInfo Cmd_ImpathKXBegin Cmd_ImpathKXFinish Cmd_ImpathReceive Cmd_ImpathSend	Support for Impath secure channel. Requires Feature Enabled.	NSO / JSO / User	R, W	KML, Impath keys
Key generation Cmd_GenerateKey Cmd_GenerateKeyPair	Generates a cryptographic key of a given type with a specified ACL. It returns a handle to the key. Optionally, it returns a KML signed certificate with the hash of the key and its ACL information.	Unauthenticated	R, W	KML, DRBG internal state, KA, KJSO,
Key import Cmd_Import	Loads a plain text key into the module.	Unauthenticated	R	KA, KJSO
Derive Key Cmd_DeriveKey	Performs key wrapping, unwrapping, transport and derivation. The ACL needs to authorize this operation.	NSO / JSO / User	R, W	KA, KJSO
Load Blob Cmd_LoadBlob	Load a Key blob into the module. It returns a handle to the key suitable for use with module services.	NSO / JSO / User	W	KA, KRE, KR, KJSO, KM, KNSO

Load Logical Token Cmd_LoadLogicalToken	Initiates loading a Logical Token from Shares, which can be loaded with the Read Share command.	Unauthenticated	-	None
Generate Logical Token Cmd_GenerateLogicalToken	Creates a new Logical Token with given properties and secret sharing parameters.	Unauthenticated	W	KM, LT, KJSO
Message digest Cmd_Hash	Computes the cryptographic hash of a given message.	Unauthenticated	-	None
Modular Exponentiation Cmd_ModExp Cmd_ModExpCrt Cmd_RSAImmedVerifyEncrypt Cmd_RSAImmedSignDecrypt	Performs a modular exponentiation (standard or CRT) on values supplied with the command.	Unauthenticated	-	None
Module hardware information Cmd_ModuleInfo	Reports detailed hardware information.	Unauthenticated	-	None
No Operation Cmd_NoOp	No operation.	Unauthenticated	-	None
Change Share Passphrase Cmd_ChangeSharePIN	Updates the passphrase of a Share.	NSO / JSO / User	R, W	Share Keys
NVRAM Allocate Cmd_NVMemAllocate	Allocation in NVRAM.	NSO / JSO	-	None
NVRAM Free Cmd_NVMemFree	Deallocation from NVRAM.	NSO / JSO	-	None
Operation on NVM list Cmd_NVMemList	Returns a list of files in NVRAM.	Unauthenticated	-	None
Operation on NVM files Cmd_NVMemOp	Operation on an NVRAM file.	Unauthenticated		None
Key export	Exports a key in plain text.	NSO / JSO / User	R	KA

Cmd_Export				
Pause for notifications Cmd_PauseForNotifications	Wait for a response from the module.	Unauthenticated		None
Read file Cmd_ReadFile	Reads data from a file on a Smartcard or Softcard. The ACL needs to authorize this operation.	NSO / JSO	-	None
Read share Cmd_ReadShare	Reads a share from a Smartcard or Softcard. Once a quorum of shares have been loaded, the module re-assembles the Logical Token.	NSO / JSO / User	R	Share Keys, LT
Send share to remote slot Cmd_SendShare	Reads a Share and encrypts it with the Impath session keys for transmission to the peer module.	NSO / JSO / User	R	Impath Keys, Share Keys
Receive share from remote slot Cmd_ReceiveShare	Receives a Share encrypted with the Impath session keys by a remote module.	NSO / JSO / User	R	Impath Keys, Share Keys
Redeem Ticket Cmd_RedeemTicket	Gets a handle in the current name space for the object referred to by a ticket created by Get Ticket.	NSO / JSO / User	-	None
Remote Administration Cmd_DynamicSlotCreateAssociation Cmd_DynamicSlotExchangeAPDUs Cmd_DynamicSlotsConfigure Cmd_DynamicSlotsConfigureQuery Cmd_VerifyCertificate	Provides remote presentation of Smartcards using a secure channel between the module and the Smartcard.	NSO / JSO / User	R, W	Remote administration session keys
Destroy Cmd_Destroy	Remove handle to an object in RAM. If the current handle is the only one remaining, the object is deleted from RAM.	Unauthenticated	Z	KA, KJSO, KNSO, LT
Report statistics Cmd StatGetValues Cmd_StatEnumTree	Reports the values of the statistics tree.	Unauthenticated	-	None
Show Status	Report status information.	Unauthenticated	-	None

Cmd_NewEnquiry				
Secure Execution Engine Cmd_CreateSEEWorld Cmd_GetWorldSigners Cmd_SEEJob Cmd_SetSEEMachine Cmd_TraceSEEWorld	Creation and interaction with SEE machines.	NSO / JSO	-	None
Set ACL Cmd_SetACL	Replaces the ACL of a given key with a new ACL. The ACL needs to authorize this operation.	NSO / JSO / User	W	KA
Set key application data Cmd_SetAppData	Writes the application information field of a key.	User	W	KA
Set NSO Permissions Cmd_SetNSOPerms	Sets the NSO key hash and which permissions require a Delegation Certificate.	NSO	-	None
Set real time clock Cmd_SetRTC	Sets the Real-Time Clock value.	NSO / JSO	-	None
Signature generation Cmd_Sign	Generate a digital signature or MAC value.	NSO / JSO / User	R	KA, KNSO, KJSO
Sign Module State Cmd_SignModuleState	Returns a signed certificate that contains data about the current configuration of the module.	Unauthenticated	R	KML
Signature verification Cmd_Verify	Verifies a digital signature or MAC value.	NSO / JSO / User	R	KA
Write file Cmd_WriteFile	Writes a file to a Smartcard or Softcard.	NSO / JSO	-	None
Write share Cmd_WriteShare	Writes a Share to a Smartcard or Softcard.	Unauthenticated	-	None

4 Physical Security

The product is a multi-chip embedded Cryptographic Module, as defined in FIPS 140-2. It is enclosed in a hard and opaque epoxy resin which meets the physical security requirements of FIPS 140-2 level 3.

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

To ensure physical security, the module should be inspected periodically for evidence of tamper attempts:

- Examine the entire PCIe board including the epoxy resin security coating for obvious signs of damage.
- Examine the heat sink on top of the module and also the potting which binds the edges of the heat sink for obvious signs of damage.
- Examine the smartcard reader and ensure it is directly plugged into the module or into the port provided by any appliance in which the module is integrated and the cable has not been tampered with.

The module has a clear button. Pressing this button puts the module into the self-test state, clearing all stored key objects, Logical Tokens and impath keys and running all self-tests. The long term security critical parameters, NSO's key, module keys and module signing key can be cleared by returning the module to the factory state.

5 Rules

This section describes how to accept, initialise and operate the module in the FIPS approved mode.

5.1 Delivery

The nShield Cryptographic Module is sent to the customers using a standard carrier service. After accepting the delivery of the module, the Crypto Officer shall perform a physical inspection of the module (refer to Physical Security). This inspection is done to ensure that the module has not been tampered with during transit. If the inspection results indicate that the module has not been tampered with, the Crypto Officer can then proceed with installation and configuration of the module.

The module must be installed and configured according to the User Guides and the Initialization procedures described below.

5.2 Initialization procedures

To configure the Cryptographic Module in FIPS approved mode, the following steps must be followed:

1. Put the module in pre-initialization mode.
2. Create a FIPS 140-2 level 2 compliant Security World using nCipher supplied utility *new-world.* Omitting the mode flag will create a Security World compliant with FIPS 140-2 Level 2.
3. Put the module in Operational mode.

An operator can verify that the module is configured in FIPS approved mode with the command line utility *nfkminfo*, which reports mode *none*.

5.3 Creation of new Operators

New User

To create a new User, the following steps must be followed:

1. Authenticate as NSO or JSO role.
2. Create a new Logical Token, LTU.
3. Split the LTU into one or more smartcards or a Softcard.
4. Generate a new Application key with the ACL configured so that the key can only be blobbed under LTU.
5. Generate a Key Blob for the Application key protected by LTU.
6. Give to the Operator the Key Blob, the Operator Cards or Softcard.

New Junior Security Officer (JSO)

To create a new JSO, the following steps must be followed:

1. Authenticate as NSO or JSO role.
2. Generate a new Logical Token, LTJSO.
3. Split LTJSO into one or more smartcards or Softcard.
4. Generate a new asymmetric key pair (KJSOpriv, KJSOpub):
    a. Set the ACL of KJSOpriv to allow Sign and UseAsSigningKey,
    b. Set the ACL of KJSOpub to allow ExportAsPlain
5. Generate a Key Blob for KJSOpriv protected by LTJSO
6. Export KJSOpub.
7. Create a Delegation Certificate signed by NSO or an already existing JSO, which includes KJSOpriv as the certifier and authorises the following actions
    a. OriginateKey, which authorises generation of new keys,
    b. GenerateLogToken, which authorises the creation of new Logical Tokens,
    c. ReadFile, WriteFile,
    d. FormatToken.
8. Give the Operator the Certificate, the Key Blob, the smartcards or Softcard.

6 Self-tests

The Cryptographic Module performs power-up and conditional self-tests. It also supports power-up self-tests upon request by resetting the module, either by pressing the Clear button or by sending the Clear Unit command.

In the event of a self-test failure, the module enters an error state which is signalled by the SOS morse pattern flashing in the output LED. While in this state, the module does not process any commands.

6.1 Power-up self-tests

In the self-test state the module clears the RAM, thus ensuring any loaded keys or authorization information is removed and then performs the following:

- Power-up self-test on hardware components,
- Firmware integrity verification,
- Cryptographic self-tests as specified below.

Table 10 Cryptographic algorithm self-tests

Algorithm	Description
AES	Known Answer Test: ECB encryption and decryption with 128, 192 and 256-bit keys
AES CMAC	Known Answer Test: 128-bit key
TDES	Known Answer Test: ECB encryption and decryption with 192-bit keys
TDES CBC MAC	Known Answer Test: 192-bit key
SHA1	SHA1 KAT test, other size are tested along with KAT HMAC
HMAC with SHA1, SHA224, SHA256, SHA384, SHA512	Known Answer Test
RSA	Known Answer Test: sign/verify, encrypt/decrypt with 2048-bit key Pair-Wise consistency test: sign/verify
DSA	Known Answer Test: sign/verify with 2048-bit key Pair-Wise consistency test: sign/verify

ECDSA	Pair-Wise consistency test: sign/verify with curves P-224 and B-233
Key Agreement	Primitive Z Known Answer Test for modular exponentiation
Key Agreement	Primitive Z Known Answer Test for point multiplication with curves P-384 and B-233
KBKDF	Known Answer Test
DRBG	Health Tests according to SP 800-90A *Note: continuous test are done on the entropy source, but are not required or useful for the DRBG, as described in IG 9.8*
Entropy source	Continuous test

6.2 Conditional self-tests

The module performs pair-wise consistency checks when RSA, DSA and ECDSA keys are generated and the continuous test on the entropy source.

6.3 Firmware load test

The Cryptographic Module supports firmware upgrades in the field, with authenticity, integrity and roll-back protection for the code. nCipher provides signed firmware images with the Firmware Integrity Key.

The module performs the following actions before replacing the current image:

- Code signature verification with the public Firmware Integrity Key.
- Image decryption with the Firmware Confidentiality Key.
- Verification that the Version Security Number (VSN) of the new image is not less than the VSN of the current image.

Note: updating the firmware to a non-FIPS validated version of the firmware will result in the module operating in a non-Approved mode.

Contact Us

Web site: https://www.ncipher.com
Help Centre: https://help.ncipher.com
Email Support: support@ncipher.com

You can also contact our Support team by telephone, using the following numbers:

Americas	Asia Pacific	Europe, Middle East and Africa
Toll Free: +1 833 425 1990 Fort Lauderdale: +1 954 953 5229	Hong Kong: +852 3461 3088 Australia: +61 8 9126 9070 Japan: +81 50 3196 4994	United Kingdom: +44 1223 723 711

www.ncipher.com

About nCipher Security

Today's fast moving digital environment enhances customer satisfaction, gives competitive advantage and improves operational efficiency. It also multiplies the security risks. nCipher Security, a leader in the general purpose hardware security module (HSM) market, empowers world-leading organizations by delivering trust, integrity and control to their business critical information and applications.

Our cryptographic solutions secure emerging technologies – cloud, IoT, blockchain, digital payments – and help meet new compliance mandates, using the same proven technology that global organizations depend on today to protect against threats to their sensitive data, network communications and enterprise infrastructure. We deliver trust for your business critical applications, ensuring the integrity of your data and putting you in complete control – today, tomorrow, at all times. www.ncipher.com

Search: nCipherSecurity

TRUST. INTEGRITY. CONTROL.

Europe, Middle East and Africa: One Station Square, Cambridge CB1 2GA • Tel: +44 1223 723 711 • Email: emea.sales@ncipher.com
Americas: Sawgrass Commerce Center – A, Suite 130, 13800 NW 14 Street, Sunrise, FL 33323 USA • Email: ams.sales@ncipher.com
Asia: 10/F, V-Point, 18 Tang Lung Street, Causeway Bay, Hong Kong • Email: apac.sales@ncipher.com
Australia: World Trade Centre Northbank Wharf, Siddeley St, Melbourne VIC 3005, Australia • Email: apac.sales@ncipher.com