CliniComp Data Acquisition Cryptographic Module

Module Version cci-das-fips-crypto-1.0

FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.2

Last update: 2023/08/25

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

List of Tables

List of Figures

Copyrights and Trademarks

CliniComp is a registered trademark of CliniComp, Intl. or its affiliates.

1 Introduction

This document is the non-proprietary FIPS 140-2 Security Policy for version cci-das-fips-crypto-1.0 of the CliniComp Data Acquisition Cryptographic Module. It contains the security rules under which the module must be operated and describes how this module meets the requirements as specified in FIPS 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 module.

1.1 Purpose of the Security Policy

There are three major reasons that a security policy is needed:

- It is required for FIPS 140 2 validation,
- It allows individuals and organizations to determine whether a cryptographic module, as implemented, satisfies the stated security policy, and
- It describes the capabilities, protection and access rights provided by the cryptographic module, allowing individuals and organizations to determine whether it will meet their security requirements.

1.2 Target Audience

This document is part of the package of documents that are submitted for FIPS 140 2 conformance validation of the module. It is intended for the following audience:

- Developers.
- FIPS 140-2 testing lab.
- The Cryptographic Module Validation Program (CMVP).
- Customers using or considering integration of CliniComp Data Acquisition Cryptographic Module.

1.3 How this Security Policy was Prepared

The vendor has provided the non-proprietary Security Policy of the cryptographic module, which was further consolidated into this document by atsec information security together with other vendor-supplied documentation as guided by FIPS 140-2 IG G.9. In preparing the Security Policy document, the laboratory formatted the vendor-supplied documentation for consolidation without altering the technical statements therein contained. The further refining of the Security Policy document was conducted iteratively throughout the conformance testing, wherein the Security Policy was submitted to the vendor, who would then edit, modify, and add technical contents. The vendor would also supply additional documentation, which the laboratory formatted into the existing Security Policy, and resubmitted to the vendor for their final editing.

2 Cryptographic Module Specification

2.1 Module Overview

The CliniComp Data Acquisition Cryptographic Module (hereafter referred to as the “module”) is a software module supporting FIPS 140-2 Approved cryptographic algorithms within CliniComp Data Acquisition Cryptographic Module. The module provides a C language application program interface (API) for use by other processes that require cryptographic functionality.

2.2 FIPS 140-2 Validation Scope

Table 1 shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard.

Table 1: FIPS 140-2 Security Requirements.

Security Requirements Section		Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles and Services and Authentication	1
4	Finite State Machine Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A
Overall Level		1

2.3 Definition of the Cryptographic Module

The CliniComp Data Acquisition Cryptographic Module is defined as a Multi-chip Standalone module per the requirements within FIPS 140-2. The logical cryptographic boundary of the module consists of shared library files and their integrity test HMAC files, which are installed by means of a full-system image per Section 10.

The CliniComp Data Acquisition Cryptographic Module package includes the binary files, integrity check HMAC files, Man Pages and the OpenSSL engines provided by the standard OpenSSL shared library. The OpenSSL engines and their shared object files are not part of the module, and therefore they must not be used when operating the module.

Table 2 defines the software components of the cryptographic module, defining its logical boundary.

Table 2: Software components of the cryptographic module.

Component	Description
/usr/lib/libcrypto.so.1.1	Shared library for cryptographic algorithms.
/usr/lib/libssl.so.1.1	Shared library for TLS network protocol.
/usr/lib/.libcrypto.so.1.1.hmac	Integrity check HMAC value for the libcrypto shared library.
/usr/lib/.libssl.so.1.1.hmac	Integrity check HMAC value for the libssl shared library.

Figure 1 shows the logical block diagram of the module executing in memory on the host system. The logical cryptographic boundary is indicated with a dashed colored box.

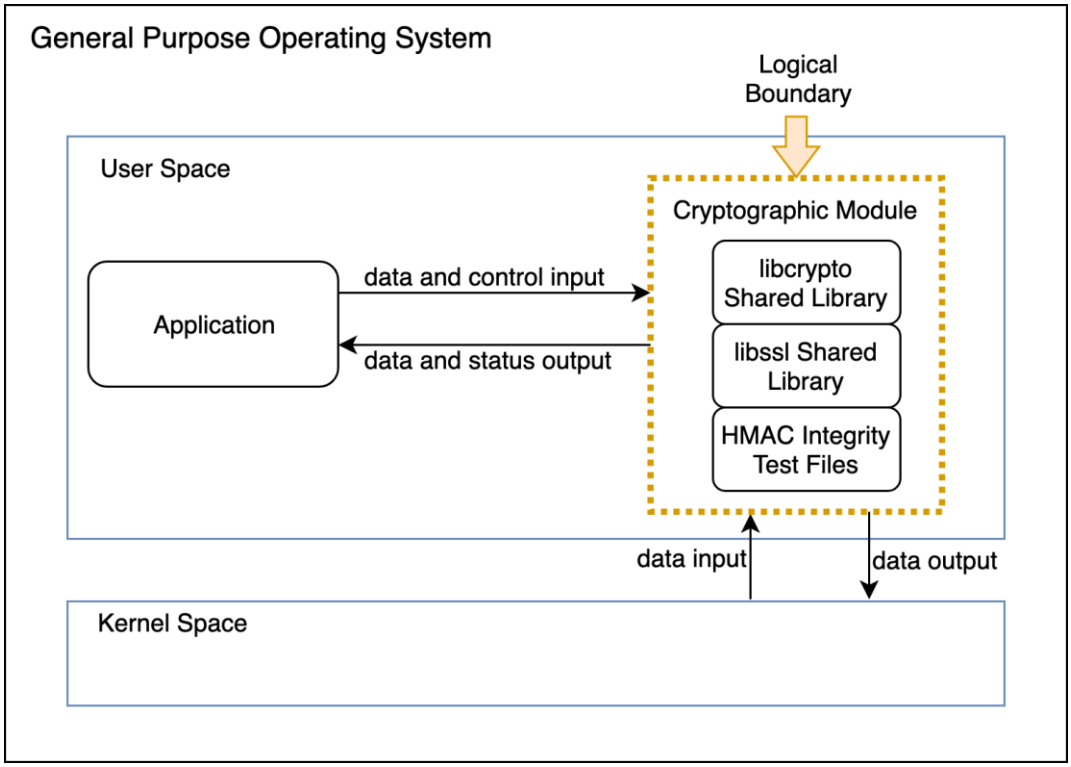

Figure 1: Logical cryptographic boundary.

2.4 Definition of the Physical Cryptographic Boundary

The physical cryptographic boundary of the module is defined as the hard enclosure of the host system on which the module runs. Figure 2 depicts the hardware block diagram. The physical hard enclosure is indicated by the yellow colored line. No components are excluded from the requirements of FIPS 140-2.

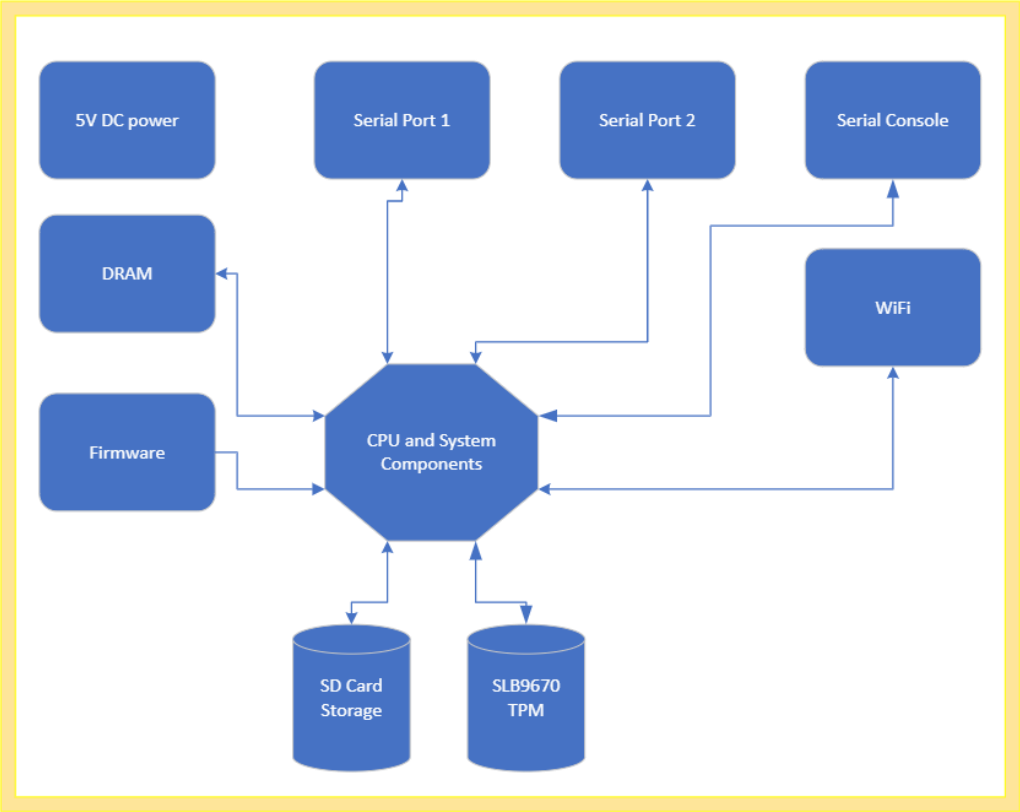

Figure 2: Hardware block diagram.

2.5 Tested Environments

The module was tested on the environments/platforms listed in Table 3. The tested operational environment is not a virtualized cloud service and was controlled such that the laboratory had full and exclusive access to the environment and module during the testing procedures.

Table 3: Tested operational environments.

Operating System	Processor	Hardware
Linux ndas-hkpg 4.19.81	Broadcom BCM2835 ARM1176jzf-s ARMv6	Raspberry Pi Zero W Rev 1.1
Linux ndas-x223 5.4.84-dey+gaca7adfe2d84	NXP i.MX6 Ultralite cortexa7t2hf ARMv7	Digi International ConnectCore 6UL SBC

2.6 Modes of Operation

The module supports two modes of operation.

- In "**FIPS mode**" (the Approved mode of operation), only approved or allowed security functions with sufficient security strength are offered by the module.
- In "**non-FIPS mode**" (the non-Approved mode of operation), non-approved security functions are offered by the module.

The module enters the operational mode after Power-On Self-Tests (POST) succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength[1] of the cryptographic keys chosen for the function or service.

If the POST or the Conditional Tests fail (Section 9), the module goes into the error state. The status of the module can be determined by the availability of the module. If the module is available, then it had passed all self-tests. If the module is unavailable, it is because any self-test failed, and the module has transitioned to the error state.

Keys and Critical Security Parameters (CSPs) used or stored in FIPS mode shall not be used in non-FIPS mode, and vice versa.

[1] See Section 5.6.1 in [SP800-57] for a definition of “security strength”.

3 Module Ports and Interfaces

As a Software module, the module does not have physical ports. The operator can only interact with the module through the API provided by the module. Thus, the physical ports within the physical boundary are interpreted to be the physical ports of the hardware platform on which the module runs and are directed through the logical interfaces provided by the software.

The logical interfaces are the API through which applications request services and receive output data through return values or modified data referenced by pointers. Table 4 summarizes the logical interfaces.

Table 4: Ports and interfaces.

Logical Interface	Description
Data Input	API input parameters for data.
Data Output	API output parameters for data.
Control Input	API function calls.
Status Output	API return codes, error message.

4 Roles, Services and Authentication

4.1 Roles

The module supports the following roles:

- **User role**: performs all services (in both FIPS mode and non-FIPS mode of operation), except module installation and configuration. This role is assumed by the calling application accessing the module.
- **Crypto Officer role**: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed depending on the service requested.

4.2 Services

The module provides services to calling applications that assume the user role, and human users assuming the Crypto Officer role. Table 5 and Table 6 depict all services, which are described with more detail in the user documentation.

The tables use the following convention when specifying the access permissions that the module has for each CSP or key.

- **Create (C):** the calling application can create a new CSP.
- **Read (R):** the calling application can read the CSP.
- **Update (U):** the calling application can write a new value to the CSP.
- **Zeroize (Z):** the calling application can zeroize the CSP.
- **N/A:** the calling application does not access any CSP or key during its operation.

For the “Role” column, U indicates the User role, and CO indicates the Crypto Officer role. An X marks which role has access to that service.

4.2.1 Services in the FIPS-Approved Mode of Operation

Table 5 provides a full description of FIPS Approved services and the non-Approved but Allowed services provided by the module in the FIPS-approved mode of operation and lists the roles allowed to invoke each service.

Table 5: Services in the FIPS-approved mode of operation.

Service	Service Description and Algorithms	Role		Keys and CSPs	Access Types
		U	CO		
Symmetric Encryption/Decryption	Encrypts or decrypts a block of data using AES.	X		AES Key	R
RSA Key Generation	Generate RSA asymmetric keys using DRBG.	X		RSA public/private keys	C, R, U
ECDSA Key Generation	Generate ECDSA asymmetric keys using DRBG.	X		ECDSA public/private keys	C, R, U
RSA Digital Signature Generation and	Sign and verify signature operations for	X		RSA public/private keys	R

Service	Service Description and Algorithms	Role		Keys and CSPs	Access Types
		U	CO		
Verification	RSA PCKS#1v1.5, PSS and X9.31.				
ECDSA Digital Signature Generation and Verification	Sign and verify signature for ECDSA.	X		ECDSA public/private keys	R
ECDSA public key validation	ECDSA	X		ECDSA public key	R
TLS Network Protocol v1.2	Provide data encryption and authentication over TLS network protocol with AES, HMAC.	X		AES key HMAC key Pre-master secret Master secret Derived AES key Derived HMAC key EC Diffie-Hellman public/private key pair RSA, ECDSA public/private keys	C, R, U
Key Derivation	Establish a TLS secure channel. KDF TLS in TLS v1.2.	X		Pre-master secret	R
				Master secret, derived AES and HMAC key	C, R, U
	KBKDF	X		Key derivation key	R
				Keying material	C, R, U
	PBKDF	X		Passphrase	R
				Master key	C, R, U
EC Diffie-Hellman Key Agreement	Establish a shared secret. KAS-SSC-ECC and KDF TLS in TLS v1.2	X		EC Diffie-Hellman public/private keys, Pre-master secret, shared secret, master secret, derived AES and HMAC keys	C, R, U
EC Diffie-Hellman Shared Secret Computation	Establish a shared secret. KAS SSC ECC	X		EC Diffie-Hellman public/private keys, shared secret	C, R, U
Key Wrapping with RSA	Encapsulates and decapsulates a key using RSA encrypt/decrypt primitives	X		RSA public/private keys	R
Key Wrapping with Symmetric Algorithms	Wrap and unwrap keys with AES-KW, AES-KWP	X		AES keys	R
Certificate Management	Management of key properties within	X		RSA, and ECDSA public/private keys	R, U

Service	Service Description and Algorithms	Role		Keys and CSPs	Access Types
		U	CO		
	certificates.			associated to an X.509 certificate	
Message Authentication Code (MAC)	Authenticate and verify authentication of data using HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	X		HMAC Key	R
	Authenticate and verify authentication of data using CMAC with AES.	X		AES Key	R
Message Digest	Hash data with SHS and SHA-3. SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	X		None	N/A
Random Number Generation	Generate random numbers based on the SP 800-90A DRBG.	X		Entropy input string and internal state	C, R, U
Other FIPS-related Services					
Show Status	Show status of the module state	X		None	N/A
Self-Test	Initiate power-on self-tests	X		None	N/A
Zeroize	Zeroize all critical security parameters	X		All keys and CSPs	Z
Module Installation	Installation of the module		X	None	N/A
Module Configuration	Configuration of the module		X	None	N/A

4.2.2 Services in the Non-FIPS-Approved Mode of Operation

Table 6 presents the services only available in non-FIPS-approved mode of operation.

Table 6: Services in the non-FIPS approved mode of operation.

Service	Service Description and Algorithms	Role		Keys	Access Types
		U	CO		
RSA key wrapping	Encrypts or decrypts using non-Approved RSA key size (Table 9)	X		RSA key pair	C, R, U
Symmetric Encryption/Decryption	Encrypts or decrypts using non-Approved algorithms listed in Table 9	X		ARIA, Blowfish, Camellia, CAST, CAST5, ChaCha20, DES, IDEA, Poly1305, RC2, RC4, RC5, Triple-DES keys	R
Digital Signature Generation and Verification	Sign or verify operations with RSA key lengths and ECDSA curves that are not listed in Table 7. SM2 algorithm.	X		RSA key < 2048 DSA keys Signature Generation with SHA-1 SM2 keys	C, R, U
Key Derivation	KDF TLS v1.0, v1.1, v1.3 (and TLS v1.2 with algorithms and keys not listed in Table 7)	X		Pre-master secret	R
		X		Derived RSA, DSA, ECDSA key	C, R, U
	KBKDF using non-approved message digest listed in Table 9	X		Key derivation key	R
		X		Keying material	C, R, U
	PBKDF with non-approved message digest listed in Table 9	X		Passphrase	R
		X		Master key	C, R, U
Diffie-Hellman Key Agreement	Establish a shared secret. KAS FFC	X		Diffie-Hellman public/private keys, shared secret	C, R, U
EC Diffie-Hellman Key Agreement	Establish a shared secret. KAS ECC	X		EC Diffie-Hellman with P-192 and any curve not listed in Table 7, shared secret	C, R, U
Asymmetric Key Generation	Generation of non-Approved RSA, DSA and ECDSA keys; SM2 keys	X		RSA key < 2048 DSA public/private keys ECDSA curves not listed in Table 7 SM2 keys	C, R, U
Random Number Generation	Generation of random numbers	X		seed, seed key	C, R, U

Service	Service Description and Algorithms	Role		Keys	Access Types
		U	CO		
	using the ANSI X9.31 PRNG				
Message Digest	Hashing using non-Approved hash (message digest) functions listed in Table 9 (GHASH, MD2, MD4, MD5, MDC2, RIPEMD, SM3, Whirlpool)	X		None	N/A
J-PAKE Key Agreement	Password authenticated key agreement using J-PAKE	X		J-PAKE key pair	C, R, U
Message Authentication Code	SipHash, Triple-DES-CMAC, Blake2, HMAC with keys of length < 112 bits	X		SipHash, Triple-DES keys, HMAC with keys < 112 bits	R
TLS network protocol v1.0, v1.1, v1.3	Any ciphersuites used in versions v1.0, v1.1, v1.3.	X		AES key Triple-DES key HMAC key Pre-master secret Master secret Derived AES key Derived Triple-DES key Derived HMAC key Diffie-Hellman, EC Diffie-Hellman public/private key pair RSA, DSA, ECDSA public/private keys	C, R, U

4.3 Algorithms

The module implements cryptographic algorithms that are used by the services provided by the module. The cryptographic algorithms that are approved to be used in the FIPS mode of operation are tested and validated by the CAVP. No parts of the Transport Layer Security (TLS) protocol, other than the key derivation function, have been tested by the CAVP or by the CMVP.

Table 7, Table 8 and Table 9 present the cryptographic algorithms in specific modes of operation. These tables include the CAVP certificates for different implementations, the algorithm name, respective standards, the available modes and key sizes wherein applicable, and usage. Information from certain columns may be applicable to more than one row.

4.3.1 FIPS-Approved

Table 7 lists the cryptographic algorithms that are approved to be used in the FIPS mode of operation.

Note: the module does not implement, in the FIPS approved mode, all algorithms for which CAVP certificates were obtained.

Table 7: FIPS-approved cryptographic algorithms.

Algorithm	Standard	Mode/Method	Key Lengths, Curves, Moduli (bits)	Use	CAVP Cert#
AES	FIPS197 SP800-38A	ECB, CBC, CFB1, CFB8, CFB128, OFB, CTR	128, 192 and 256	Data Encryption and Decryption	A802
	FIPS197 SP800-38B	CMAC	128, 192 and 256	MAC Generation and Verification	
	FIPS197 SP800-38C FIPS197	CCM	128, 192 and 256	Data Encryption and Decryption	
	FIPS197 SP800-38E	XTS	128 and 256	Data Encryption and Decryption	
	FIPS197 SP800-38F	KW, KWP	128, 192 and 256	Key Wrapping	
	FIPS197 SP800-38D	GCM with external IV, mode 8.2.1	128, 192 and 256	Data Encryption and Decryption	A801
		GCM with internal IV, mode 8.2.1	128, 192 and 256	Data Encryption	A801
DRBG	SP800-90A	CTR_DRBG AES128, AES192, AES256 with/without DF, with/without PR	n/a	Random Number Generation	A802[2]
ECDSA	FIPS186-4	Testing candidates	P-224	Key Pair Generation	A863
			P-256, P-384, P-521		A800
		SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521	Digital Signature Generation	A800
			P-224	Public Key Verification	A863
			P-256, P-384, P-521		A800

[2] This same implementation of DRBG was tested by itself with Cert. A803.

Algorithm	Standard	Mode/Method	Key Lengths, Curves, Moduli (bits)	Use	CAVP Cert#
		SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	P-224, P-256, P-384, P-521	Digital Signature Verification	A800
KAS-ECC-SSC SP800-56Ar3	SP800-56Arev3 IG D.8	ECC Ephemeral Unified scheme	P-224, P-256, P-384, P-521	Shared Secret Computation	A800
KAS	IG D.8 Scenario X1 (2)	KAS-ECC-SSC and KDF TLS Component	P-256, P-384, P-521	Key Agreement	A800 (KAS-ECC-SSC and KDF TLS)
HMAC	FIPS198-1	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	112 or greater	Message Authentication Code	A800
		SHA3-224, SHA3-256, SHA3-384, SHA3-512	112 or greater	Message Authentication Code	A799
KDF TLS Component	SP800-135	TLS v1.2, SHA2-256, SHA2-384		Key Derivation	A800
PBKDF	SP800-132	HMAC-[SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512]	112 or greater	Key Derivation	A800
		HMAC-[SHA3-224, SHA3-256, SHA3-384, SHA3-512]	112 or greater	Key Derivation	A799
RSA	FIPS186-4	B.3.3 Random Probably Primes	2048, 3072, 4096	Key Pair Generation	A800
		X9.31 with SHA2-256, SHA2-384, SHA2-512	2048, 3072, 4096	Digital Signature Generation	
		PKCS#1v1.5 and PSS with SHA2-224, SHA2-256, SHA2-384, SHA2-512	2048, 3072, 4096		
		X9.31 with SHA-1, SHA2-256,	1024, 2048, 3072, 4096	Digital Signature Verification	

Algorithm	Standard	Mode/Method	Key Lengths, Curves, Moduli (bits)	Use	CAVP Cert#
		SHA2-384, SHA2-512			
		PKCS#1v1.5 and PSS with SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	1024, 2048, 3072, 4096		
SHS	FIPS180-4	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512	N/A	Message Digest	A800
SHA-3	FIPS202	SHA3-224, SHA3-256, SHA3-384, SHA3-512	N/A	Message Digest	A799
		SHAKE-128, SHAKE-256	N/A	Extendable Output Function	A799
KBKDF	SP800-108	Counter mode and Feedback mode with HMAC-[SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512] Counter mode and Feedback mode with CMAC-[AES128, AES192, AES256]	8, 72, 128, 776, 3456, 4096	Key Derivation	A804
KTS	SP800-38F	AES-KW, KWP	128, 192, 256	Key Wrapping	A802
		AES-GCM, AES-CCM	128, 256	Key Wrapping	A801 A802
		AES-CBC and HMAC-SHA2	128, 256	Key Wrapping	A802 A800

4.3.2 Non-Approved-but-Allowed

Table 8 lists the non-Approved-but-Allowed cryptographic algorithms provided by the module that are allowed to be used in the FIPS mode of operation.

Table 8: Non-Approved-but-allowed cryptographic algorithms.

Algorithm	Usage
RSA Key Wrapping with key size between 2048 bits and 15360 bits or more	Key wrapping, key establishment methodology provides between 112 and 256 bits of encryption strength.
NDRNG	Used for seeding NIST SP 800-90A DRBG.

4.3.3 Non-Approved

Table 9 lists the cryptographic algorithms that are not allowed to be used in the FIPS mode of operation. Use of any of these algorithms (and corresponding services in Table 6) will implicitly switch the module to the non-Approved mode.

Table 9: Non-FIPS approved cryptographic algorithms.

Algorithm	Usage
ARIA	Data encryption and decryption
Blowfish	Data encryption and decryption
Camellia	Data encryption and decryption
CAST, CAST5	Data encryption and decryption
ChaCha20	Data encryption and decryption
DES	Data encryption and decryption
IDEA	Data encryption and decryption
Poly1305	Data encryption and decryption
RC2	Data encryption and decryption
RC4	Data encryption and decryption
RC5	Data encryption and decryption
Triple-DES	Data encryption and decryption
SM2	Digital signature generation and verification
Diffie-Hellman	Key agreement
EC Diffie-Hellman	Key agreement using curves not listed in Table 7
SRP	Key agreement.
KDF TLS for TLS 1.0, 1.1, 1.3	Key derivation
PBKDF	Key derivation using message digest algorithms not listed in Table 7

Algorithm	Usage
ECDSA	Key generation/signature generation with curves not listed in Table 7
RSA	Key generation/Signature generation with keys of length less than 2048 bits.
SipHash	Message authentication code
Triple-DES-CMAC	Message authentication code
HMAC	Message authentication code with keys of length less than 112 bits
Blake2	Message digest
GHASH	Message digest
MD2	Message digest
MD4	Message digest
MD5	Message digest
MDC2	Message digest
RIPEMD	Message digest
SM3	Message digest
Whirlpool	Message digest
DSA	Parameter/key generation/signature generation
J-PAKE	Password Authenticated Key Exchange
ANSI X9.31 RNG	Random number generation
SHA-1	Signature generation

4.4 Operator Authentication

The module does not support operator authentication mechanisms. The role of the operator is implicitly assumed based on the service requested.

5 Physical Security

The module is comprised of software only and thus this Security Policy does not claim any physical security.

6 Operational Environment

6.1 Applicability

The module operates in a modifiable operational environment per FIPS 140-2 Security Level 1 specifications. The module runs on the Linux ndas-hkpg 4.19.81 and the Linux ndas-x223 5.4.84-dey+gaca7adfe2d84 operating systems executing on the hardware specified in Section 2.5.

6.2 Policy

The operating system is restricted to a single operator mode of operation; concurrent operators are explicitly excluded by the operating system.

The application that makes calls to the module is the single user of the module, even when the application is serving multiple clients.

The ptrace system call, the debugger gdb and strace shall not be used. In addition, other tracing mechanisms offered by the Linux environment, such as ftrace or systemtap shall not be used.

7 Cryptographic Key Management

Table 10 summarizes the keys and other CSPs that are used by the cryptographic services implemented in the module.

Table 10: Lifecycle of keys and other Critical Security Parameters (CSPs).

Name	Use	Generation	Entry and Output	Zeroization
AES key	Encryption, decryption. MAC generation and verification for CMAC. Key wrapping.	Provided by the calling application.	Entered via API input parameter. No output.	EVP_CIPHER_CTX_free(), EVP_CIPHER_CTX_reset()
AES derived key	Encryption, decryption. MAC generation and verification for CMAC.	Generated internally by the module (from the SP800-135 KDF TLS) during the establishment of the TLS tunnel.	N/A	EVP_CIPHER_CTX_free(), EVP_CIPHER_CTX_reset()
HMAC key	MAC generation and verification.	Provided by the calling application.	Entered via API input parameter. No output.	HMAC_CTX_free()
HMAC derived key	MAC generation and verification.	Generated internally by the module (from the SP800-135 KDF TLS) during the establishment of the TLS tunnel.	N/A	HMAC_CTX_free()
RSA public and private key	RSA signature generation and verification. Key wrapping.	Keys are generated using FIPS 186-4 and the random value used in the key generation is obtained from SP800- 90A DRBG.	Entered via API input parameter or generated by module. Output via API output parameters in plaintext.	RSA_free()
ECDSA public and private key	ECDSA signature generation and verification.	Keys are generated using FIPS 186-4 and the random value used in	Entered via API input parameter or generated by module.	EC_KEY_free()

Name	Use	Generation	Entry and Output	Zeroization
		the key generation is obtained from SP800- 90A DRBG.	Output via API output parameters in plaintext.	
EC Diffie-Hellman public and private key	Key agreement.	Keys are generated using FIPS 186-4 and the random value used in the key generation is obtained from SP800- 90A DRBG.	Entered via API input parameter or generated by module. Output via API output parameters in plaintext.	EC_KEY_free()
Pre-master secret (shared secret)	Establishment of encrypted session.	Generated during the key agreement when using EC Diffie-Hellman key exchange.	Entry: No entry. Output: No output.	SSL_free(), SSL_clear()
Shared secret	Establishment of derived keys.	Generated during the key agreement when using EC Diffie-Hellman key exchange.	Output via API output parameters in plaintext.	EC_KEY_free()
Passphrase	Entry for PBKDF.	N/A	Entered via API input parameters. No output.	EVP_PKEY_free(), kdf_pbkdf2_free()
Master key	Cryptographic key (output from PBKDF)	Derived from passphrase by PBKDF.	Output via API output parameters in plaintext.	EVP_PKEY_free()
Key derivation key	Cryptographic key (input for KBKDF).	N/A	Entered via API input parameters. No output.	EVP_PKEY_free()
Keying material (KBKDF)	Cryptographic key (output from KBKDF).	Derived from key derivation key by KBKDF.	Output via API output parameters in plaintext.	EVP_PKEY_free()
Master secret	Establishment of encrypted session.	Derived from pre-master secret.	N/A	SSL_free(), SSL_clear()

Name	Use	Generation	Entry and Output	Zeroization
Entropy input string	Entropy input strings used to compose seed to the DRBG.	Obtained from NDRNG.	N/A	FIPS_drbg_free()
DRBG internal state (V, Key)	V and key are used internally by CTR DRBGs. Used to generate random bits.	During DRBG initialization.	N/A	FIPS_drbg_free()

7.1 Random Number Generation

The module provides a DRBG compliant with SP800-90A for the creation of key components of asymmetric keys, and random number generation. The DRBG implements a CTR_DRBG mechanism. The DRBG is initialized during module initialization; the module loads by default the DRBG using the CTR_DRBG mechanism with AES-256, with derivation function, and without prediction resistance. A different DRBG mechanism can be chosen through an API function call.

The module uses a Non-Deterministic Random Number Generator (NDRNG) to seed the DRBG through a call to the getrandom() system function. The NDRNG is provided by the operational environment (i.e., Linux RNG), which is within the module's physical boundary but outside of the module's logical boundary. The NDRNG provides at least 128 bits of entropy to construct the seed for seeding and reseeding of the DRBG.

The Linux kernel performs conditional self-tests on the output of NDRNG to ensure that consecutive random numbers do not repeat. The module performs the DRBG health tests as defined in section 11.3 of SP800-90A.

7.2 Key/CSP Generation

The module provides an SP800-90A-compliant Deterministic Random Bit Generator (DRBG) for creation of key components of asymmetric keys, and random number generation.

For generating RSA, ECDSA, and EC Diffie-Hellman keys, the module implements asymmetric key generation services compliant with FIPS186-4 and SP800-56Arev3, and using the DRBG compliant with SP800-90A. The random value used in asymmetric key generation is obtained from the DRBG. In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per SP800-133 (vendor affirmed).

Symmetric keys are derived from the shared secret established by EC Diffie- Hellman in a manner that is compliant to NIST SP 800-135 for KDF TLS, and from the KBKDF and PBKDF algorithms.

The module does not support manual key entry or intermediate key generation output. In addition, the module does not produce key output outside its physical boundary. The keys can be entered or output from the module in plaintext form via API parameters, to and from the calling application only.

The module generates cryptographic keys whose strengths are modified by available entropy.

7.3 Key/CSP Storage

Symmetric keys, public and private keys are provided to the module by the calling process and are destroyed when released by the appropriate API function calls.

The module does not perform persistent storage of keys. The keys and CSPs are stored as plaintext in the RAM. The only exception is the HMAC key used for the integrity test, which is stored in the module and relies on the operating system for protection.

7.4 Key/CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The application is responsible for calling the appropriate zeroization functions provided in the module's API and listed in Table 10. Calling the SSL_free() and SSL_clear() will zeroize the keys and CSPs stored in the TLS protocol internal state and also invoke the corresponding API functions listed in Table 10 to zeroize keys and CSPs. The zeroization functions overwrite the memory occupied by keys with “zeros” and deallocate the memory with the regular memory deallocation operating system call.

7.5 Key Establishment

The module provides EC Diffie-Hellman key agreement schemes from SP800-56Arev3. These key agreement schemes are used as part of the TLS protocol key exchange.

The module provides AES key wrapping approved by SP800-38F and RSA key wrapping (encapsulation) using public key encryption and private key decryption primitives as allowed by FIPS140-2 IG D.9. RSA key wrapping may be used as part of the TLS protocol key exchange.

The module provides approved key transport methods according to IG D.9. The key transport methods are provided either by:

- Using an approved key wrapping algorithm (AES-KW, AES-KWP).
- An approved authenticated encryption mode (AES-GCM, AES-CCM).
- A combination method that occurs exclusively within the context of a TLS protocol connection, using TLS ciphersuites. The combination method consists of using an approved symmetric encryption mode (AES-128-CBC, AES-256-CBC) together with an approved authentication method (HMAC-SHA2), again within the context of the TLS protocol ciphersuites.

Table 7 and Table 8 specify the key sizes allowed in the FIPS mode of operation. According to Table 2 in SP800-57, the key sizes of key wrapping and transport, RSA, and EC Diffie-Hellman provide the following security strengths:

- AES key wrapping provides between 128 and 256 bits of encryption strength.
- RSA key wrapping provides between 112 and 256 bits of encryption strength.
- EC Diffie-Hellman key establishment methodology provides between 112 and 256 bits of encryption strength.
- Approved authenticated encryption mode key establishment methodology (AES-GCM, AES-CCM) provides 128 or 256 bits of encryption strength.
- Combination of approved AES-128-CBC and AES-256-CBC encryption and HMAC-SHA2 authentication key establishment methodology provides 128 or 256 bits of encryption strength.

The module supports the following key derivation functions:

- KDF TLS for TLS v1.2.
- KBKDF per SP800-108.
- PBKDF compliant with option 1(a) of SP800-132. Keys derived from passwords or passphrases using this method can only be used in storage applications.

The module does not support manual key entry or intermediate key generation key output. The keys are provided to the module via API input parameters in plaintext form and output via API output parameters in plaintext form. This is allowed by IG 7.7, according to the “CM Software to/from App Software via GPC INT Path” entry on the Key Establishment Table.

8 Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms listed in Table 3 have been tested and found to conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, FCC PART 15, Subpart B, Unintentional Radiators, Digital Devices, Class B (i.e., home use). These devices are designed to provide reasonable protection against harmful interference when the devices are operated in a commercial environment.

9 Self-Tests

9.1 Power-On Self-Tests

The module performs power-on self-tests (POSTs) automatically during loading of the module by making use of default entry point (DEP). These POSTs ensure that the module is not corrupted and that the cryptographic algorithms work as expected. No operator intervention is necessary to run the POSTs.

While the module is executing the POSTs, services are not available, and input and output are inhibited. The module is not available for use until successful completion of the POSTs.

The integrity of the module binary is verified using an HMAC-SHA2-256. The HMAC value is computed at build time and stored in the .hmac file. The value is recalculated at runtime and compared against the stored value in the file. If the comparison succeeds, then the remaining POSTs (consisting of the algorithm-specific Known Answer Tests) are performed. On successful completion of the all the power-up tests, the module becomes operational and cryptographic services are then available. If any of the tests fails, the module transitions to the error state and subsequent calls to the module will fail. Thus, in the error state, no further cryptographic operations will be possible.

Table 11 details the self-tests that are performed on the FIPS-approved cryptographic algorithms supported in the FIPS-approved mode of operation, using the Known-Answer Tests (KATs) and Pairwise Consistency Tests (PCTs).[3]

Table 11: Self-tests.

Algorithm	Test
AES	• KAT AES-ECB with 128-bit key, encryption and decryption • KAT AES-CCM with 192-bit key, encryption and decryption • KAT AES-GCM with 256-bit key, encryption and decryption • KAT AES-CMAC with 128-bit, 192-bit and 256-bit key, MAC generation and verification • KAT AES-XTS with 128-bit and 256-bit keys, encryption and decryption
RSA[4]	• KAT RSA PKCS#1v1.5 signature generation and verification with 2048-bit key and SHA2-256 • KAT RSA PSS signature generation and verification with 2048-bit key and SHA2-256 • KAT RSA with 2048-bit key, public-key encryption and private key decryption

[3] Note: the module uses IG 9.1 and IG 9.4 to fulfill the requirements for self-tests for all algorithms. For instance, the AES-GCM and AES-ECB self-tests fulfill the requirements for self-tests for all AES modes.

[4] Note: Although the self-test for RSA encrypt/decrypt are performed, it is not listed because it is a non-approved algorithm.

Algorithm	Test
ECDSA	• PCT ECDSA with P-256 and SHA2-256
KAS ECC (EC Diffie-Hellman)	• Primitive "Z" Computation KAT with P-256 curve
DRBG	• KAT CTR_DRBG using AES-256 with and without DF, with and without PR, and health tests prescribed by Section 11.3 of SP800-90A
HMAC	• KAT HMAC-SHA-1, HMAC-SHA2-224, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512
SHS	• KAT SHA-1, SHA2-256, SHA2-512
SHA-3	• KAT SHA3-256, SHA3-512, SHAKE-128 and SHAKE-256
KBKDF	• KAT using test vectors from RFC8809, HMAC-SHA2-256
PBKDF	• KAT using passphrase, salt, and HMAC-SHA2-256
KDF TLS	• KAT for KDF TLS (TLS v1.2), HMAC-SHA2-256
Module Integrity	• HMAC-SHA2-256

9.2 Conditional Self-Tests

Conditional tests are performed during operational state of the module when the respective crypto functions are used. If any of the conditional tests fails, module transitions to the error state. When the module is in the error state, no data is output, and cryptographic operations are not allowed.

Table 12 lists the conditional self-tests performed by the module.

Table 12: Conditional self-tests.

Algorithm	Test
ECDSA Key generation	PCT using SHA2-256, signature generation and verification
RSA Key generation	PCT using SHA2-256, signature generation and verification[5]

9.3 On-Demand self-tests

The module provides the Self-Test service to perform self-tests on demand. On demand self-tests can be invoked by powering-off and reloading the module. This service performs the same cryptographic algorithm tests executed during power-up. During the execution of the on-demand self-tests, cryptographic services are not available, and no data output or input is possible.

[5] Note: Although the self-test for RSA encrypt/decrypt are performed, it is not listed because it is a non-approved algorithm.

10 Guidance

This section provides guidance for the Crypto Officer and the User to maintain proper use of the module per FIPS 140-2 requirements.

10.1 Crypto-Officer Guidance

The nDAS firmware image is built from the CliniComp Embedded Engineering Git Repository and released according to the CliniComp Embedded Engineering Release Process. Only firmware released and tagged with a DAS release tag shall be installed on the nDAS. As of this writing, the release tag is das-9.1.3.

The initial SD card shall be written using the ***mkndas-sd*** script. The ***mkndas-sd*** script is part of the Embedded Engineering release package. It is installed on the server under /eerun/ndas-mfg/bin.

The default manufacturing values can be supplied in /eerun/ndas-mfg/conf/ndas-mfg.rcf. They may also be passed to mkndas-sd on the command line.

Typically, the user would specify the hardware (MAC) address and the device ID on the command line. For example:

```
/eerun/ndas-mfg/bin/mkndas-sd -m 00:40:A5:54:55:66 -u dev1
```

When the nDAS image is produced using ***mkndas-sd***, the nDAS will boot into FIPS mode automatically. To disable FIPS mode, a developer must alter the nDAS firmware image. Doing so will result in a product that is no longer the validated module described in this document. In this case, to securely re-establish the product to the validated module, the Crypto Office must reinstall the module precisely as described in this section.

10.2 User Guidance

As specified in Section 2.6, the mode of operation of this module is implicitly selected depending upon which security functions or services and key sizes or curves are being used. To run the module in FIPS mode, the user shall only use the FIPS approved or allowed services listed in Table 5, or the validated or allowed cryptographic algorithms and security functions listed in Table 7 and Table 8.

The function calls FIPS_mode_set(0), ENGINE_register_* and ENGINE_set_default_* are prohibited while running the module.

10.2.1 TLS

The TLS protocol implementation provides both the server and the client sides. In order to operate in FIPS mode, digital certificates used for server and client authentication shall comply with the restrictions of key size and message digest algorithms imposed by SP800-131A.

10.2.2 Random Number Generator

The OpenSSL API call of RAND_cleanup must not be used. This call will clean up the internal DRBG state. This call also replaces the DRBG instance with the non-FIPS Approved SSLeay Deterministic Random Number Generator when using the RAND_* API calls.

10.2.3 AES GCM IV

AES GCM encryption and decryption are used in the context of the TLS protocol version 1.2, compliant to Scenario 1 of IG A.5. The module is compliant with SP 800-52 and the mechanism for IV generation is compliant with RFC5288. The operations of one of the two parties involved in the TLS key establishment scheme are performed entirely within the cryptographic boundary of the module, including the setting of the counter portion of the IV.

The nonce_explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce_explicit, or counter portion, of the IV will not exhaust all of its possible values. This behavior complies with Section 7.4.1.1 and Section 7.4.1.2 in RFC5246 and also compliant to FIPS140-2 IG A.5.

In case the module's power is lost and then restored, the key used for AES GCM encryption or decryption shall be re-distributed.

10.2.4 AES-XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in SP800-38E. In addition, the length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks, that is, 16 MiB of data.

In addition, to meet the requirement in FIPS140-2 IG A.9, the module implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

10.2.5 Key Usage and Management

In general, a single key shall be used for only one purpose (e.g., encryption, integrity, authentication, key wrapping, random bit generation, or digital signatures) and be disjoint between the modes of operations of the module. Thus, if the module is switched between its FIPS mode and non-FIPS mode or vice versa, the following procedures shall be observed:

- The DRBG engine shall be reseeded.
- CSPs and keys shall not be shared between security functions of the two different modes.

10.3 Handling Self-Test Errors

The module transition to error state when any of self-tests or conditional tests fails. The application must be restarted to recover from these errors. Following are the error messages specific to self-test failure:

FIPS_R_FINGERPRINT_DOES_NOT_MATCH – The integrity verification check failed

FIPS_R_SELFTEST_FAILED – a known answer test failed

FIPS_R_TEST_FAILURE – a known answer test failed (RSA); pairwise consistency test failed (DSA)

FIPS_R_PAIRWISE_TEST_FAILED – a pairwise consistency test failed during EC/DSA or RSA key generation

FIPS_R_DRBG_STUCK – the DRBG generated two same consecutive values

These errors are reported through the regular ERR interface of the module and can be queried by functions such as ERR_get_error(). See the OpenSSL manual page for the function description.

When the module is in error state, output is inhibited and no crypto operations are available. Any calls to the crypto functions in error state will return error message: 'FATAL FIPS SELFTEST FAILURE' on stderr and the application is terminated with the abort() call.

The only way to recover from the error state is to reload the module and restart the application. If failures persist, the module must be reinstalled.

10.4 API Functions

Passing “0” to the FIPS_mode_set() API function is prohibited.

Executing the CRYPTO_set_mem_functions() API function is prohibited as it performs like a null operation in the module.

10.5 Key derivation using SP800-132 PBKDF

The module provides password-based key derivation (PBKDF), compliant with SP800-132. The module supports option 1(a) from Section 5.4 of SP800-132, in which the Master Key (MK) or a segment of it is used directly as the Data Protection Key (DPK).

In accordance to SP800-132 and IG D.6, the following requirements shall be met.

- Derived keys shall only be used in storage applications. The Master Key (MK) shall not be used for other purposes. The length of the MK or DPK shall be of 112 bits or more.
- A portion of the salt, with a length of at least 128 bits, shall be generated randomly using the SP800-90A DRBG,
- The iteration count shall be selected as large as possible, as long as the time required to generate the key using the entered passphrase is acceptable for the users. The minimum value shall be 1000.
- Passwords or passphrases, used as an input for the PBKDF, shall not be used as cryptographic keys.
- The length of the password or passphrase shall be of at least 20 characters, and shall consist of lower-case, upper-case, and numeric characters. The probability of guessing the value is estimated to be $1/62^{20} = 10^{-36}$, which is less than 2^{-112}.

The calling application shall also observe the rest of the requirements and recommendations specified in SP800-132.

11 Mitigation of Other Attacks

The module does not claim any mitigation of other attacks.

12 Acronyms, Terms and Abbreviations

Term	Definition
AES	Advanced Encryption Standard
AESNI	Advanced Encryption Standard New Instructions
AVX	Advanced Vector Extensions
AVX2	Advanced Vector Extensions 2
CAVP	Cryptographic Algorithm Validation Program
CMVP	Cryptographic Module Validation Program
CSE	Communications Security Establishment
CSP	Critical Security Parameter
DH	Diffie-Hellman
DHE	Diffie-Hellman Ephemeral
DRBG	Deterministic Random Bit Generator
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EDC	Error Detection Code
HMAC	(Keyed) Hash Message Authentication Code
IKE	Internet Key Exchange
KAT	Known Answer Test
KDF	Key Derivation Function
NDRNG	Non-Deterministic Random Number generator
NIST	National Institute of Standards and Technology
PAA	Processor Algorithm Acceleration
POST	Power On Self Test
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
PUB	Publication
SHA	Secure Hash Algorithm
SSSE3	Supplemental Streaming SIMD Extensions 3
VPAES	AES with Vector Permutations
TLS	Transport Layer Security

13 References

Barker, E., & Roginsky, A. (2015, 11). SP 800-131A Revision 1. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. *SP 800-131A Revision 1. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*. National Institute of Standards & Technology. Retrieved from http://dx.doi.org/10.6028/NIST.SP.800-131Ar1

Dang, Q. (2011, 12). SP 800-135 Revision 1. Recommendation for Existing Application-Specific Key Derivation Functions. *SP 800-135 Revision 1. Recommendation for Existing Application-Specific Key Derivation Functions*. National Institute of Standards & Technology. doi:https://doi.org/10.6028/NIST.SP.800-135r1

Dierks, T., & Rescorla, E. (2008, 8). The Transport Layer Security (TLS) Protocol Version 1.2. *The Transport Layer Security (TLS) Protocol Version 1.2*. Retrieved from http://tools.ietf.org/html/rfc5246

Dworkin, M. J. (2001, 12). SP 800-38A. Recommendation for Block Cipher Modes of Operation: Methods and Techniques. *SP 800-38A. Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. National Institute of Standards & Technology. doi:https://doi.org/10.6028/NIST.SP.800-38A

Dworkin, M. J. (2004, 5). SP 800-38C. Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality. *SP 800-38C. Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*. doi:https://doi.org/10.6028/NIST.SP.800-38C

Dworkin, M. J. (2005, 5). SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. *SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. doi:https://doi.org/10.6028/NIST.SP.800-38B

Dworkin, M. J. (2007). SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. *SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Gaithersburg, MD, United States: National Institute of Standards & Technology. doi:https://doi.org/10.6028/NIST.SP.800-38D

Dworkin, M. J. (2010, 1). SP 800-38E. Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices. *SP 800-38E. Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices*. Gaithersburg, MD, United States: National Institute of Standards & Technology. doi:https://doi.org/10.6028/NIST.SP.800-38E

Dworkin, M. J. (2012, 12). SP 800-38F. Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping. *SP 800-38F. Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*. Gaithersburg, MD, United States: National Institute of Standards & Technology. Retrieved from https://csrc.nist.gov/publications/detail/sp/800-38f/final

FIPS PUB 140-2. Security Requirements for Cryptographic Modules. (2001, 5 25). *FIPS PUB 140-2. Security Requirements for Cryptographic Modules*. Retrieved from https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

FIPS PUB 180-4. Secure Hash Standard (SHS). (2012, 3). *FIPS PUB 180-4. Secure Hash Standard (SHS)*. Gaithersburg, MD 20899-8900: National Institute of Standards & Technology. Retrieved from http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

FIPS PUB 186-4. Digital Signature Standard (DSS). (2013, 7). *FIPS PUB 186-4. Digital Signature Standard (DSS)*. https://doi.org/10.6028/NIST.FIPS.186-4. doi:https://doi.org/10.6028/NIST.FIPS.186-4

FIPS PUB 198-1. The Keyed-Hash Message Authentication Code (HMAC). (2008, 7). *FIPS PUB 198-1. The Keyed-Hash Message Authentication Code (HMAC)*. Retrieved from http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program. (2020, 8 28). *Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program*. Retrieved 09 22, 2020, from https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/fips1402ig.pdf

Salowey, J., Choudhury, A., & McGrew, D. (2008, 8). AES Galois Counter Mode (GCM) Cipher Suites for TLS. *AES Galois Counter Mode (GCM) Cipher Suites for TLS*. doi:https://tools.ietf.org/html/rfc5288

Turan, M. S., Barker, E., Burr, W., & Chen, L. (2010, 12). SP 800-132. Recommendation for Password-Based Key Derivation: Part 1: Storage Applications. *SP 800-132. Recommendation for Password-Based Key Derivation: Part 1: Storage Applications*. National Institute of Standards & Technology. doi:https://doi.org/10.6028/NIST.SP.800-132

The FIPS 140-2 standard, and information on the CMVP, can be found at http://csrc.nist.gov/groups/STM/cmvp/index.html. More information describing the module can be found on the vendor web site at https://www.clinicomp.com.

This Security Policy contains non-proprietary information. All other documentation submitted for FIPS 140-2 conformance testing and validation is proprietary and is releasable only under appropriate non-disclosure agreements