FIPS 140-2 Non-Proprietary Security Policy for

Unbound Tech EKM Cryptographic Module

Software Version 2.0

Document Version 005

23 April 2020

Prepared for Unbound Tech by

Rycombe Consulting Limited
http://www.rycombe.com
+44 1273 476366

Table of Contents

Table of Figures

1 Introduction

This section identifies the cryptographic module; describes the purpose of this document; provides external references for more information; and explains how the document is organized.

Unbound Tech randomly splits keys across servers so that they are never in any single place to be stolen. The advanced protocols used in Unbound Tech ensure that even if servers are breached and completely controlled by an attacker, the secrets and credentials cannot be stolen. The result is that digital assets remain safe, even if all else fails and attackers get inside the network.

Furthermore, Unbound Tech frequently refreshes the random split key process - distributing different, random key parts to each Unbound Tech Server. As result of the refresh, even in the extremely unlikely case that an attacker breaches the server and steals a key part, the key part alone is useless and it becomes obsolete as soon as the next refresh takes place. This provides a very high level of security, and enables enterprise servers to be used with a much lower level of risk.

Unbound Tech is able to protect all types of standard cryptographic keys: RSA and ECC keys for all purposes - encryption/decryption, digital signing and authentication. Unbound Tech's technology for securing keys using multi-party computation (MPC) is fully transparent to the calling application.

1.1 Identification

Module Name Unbound Tech EKM Cryptographic Module

Module Version 2.0

1.2 Purpose

This is the non-proprietary FIPS 140-2 Security Policy for the Unbound Tech EKM Cryptographic Module, also referred to as "the module" within this document. This Security Policy details the secure operation of Unbound Tech EKM Cryptographic Module as required in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

1.3 References

For more information on Unbound Tech products please visit: www.unboundtech.com. For more information on NIST and the Cryptographic Module Validation Program (CMVP), please visit http://csrc.nist.gov/groups/STM/cmvp/index.html.

1.4 Document Organization

This document is Copyright 2020 Unbound Tech Security Ltd. It may be freely reproduced and distributed whole and intact including this copyright notice. This Security Policy document is one part of the FIPS 140-2 Submission Package. This document outlines the functionality provided by the module and gives high-level details on the means by which the module satisfies FIPS 140-2 requirements. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Submission documentation may be Unbound Tech proprietary or otherwise controlled and releasable only under

appropriate non-disclosure agreements. For access to these documents, please contact Unbound Tech.

The various sections of this document map directly onto the sections of the FIPS 140-2 standard and describe how the module satisfies the requirements of that standard.

1.5 Document terminology

The following abbreviations are used in this document:

Term	Description
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Programming Interface
BIOS	Basic Input Output Services
CAVP	Cryptographic Algorithm Validation Program
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CMSP	Cryptographic Module Security Policy
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit (Microprocessor)
CSP	Critical Security Parameters
DES	Data Encryption Standard
DRBG	Deterministic Random-bit Generator
EC DH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
HMAC	Keyed-Hash Message Authentication Code
IV	Initialization Vector
KDF	Key Derivation Function
KW	Key Wrap
KWP	Key Wrap with Padding
MAC	Message Authentication Code
MPC	Secure multi-party computation
N/A	Not Applicable
NDRNG	Non-deterministic Random Number Generator
NIST	National Institute of Standards and Technology
OAEP	Optimal asymmetric encryption padding
OFB	Output Feedback
OS	Operating System
RAM	Random-access Memory
RBG	Random Bit Generator
RFC	Request for Comments
RNG	Random Number Generator
RSA	An algorithm for public-key cryptography. Named after Rivest, Shamir and Adleman who first publicly described it.
RSA PKCS1.5	RSA Public-Key Cryptography Standards (PKCS) #1 v1.5
RSA PSS	Provable secure RSA Signatures
RSADP	RSA Decryption Primitive

Term	Description
RWE	Read, Write, Execute
SCSI	Small Computer System Interface
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SIV	Synthetic IV
SP	NIST Special Publication document
TLS	Transport Layer Security
Triple-DES	Triple-DES
USB	Universal Serial Bus

Figure 1 Document Terminology

2 Unbound Tech EKM Cryptographic Module

This section provides the details of how the module meets the FIPS 140-2 requirements.

2.1 Overview

The module provides cryptographic services to Unbound Tech products.

2.2 Module Specification

The Unbound Tech EKM Cryptographic Module is a software module that provides cryptographic services to Unbound Tech products.

The module is classified as a multi-chip standalone module.

The module provides a number of NIST validated cryptographic algorithms. The module provides applications with a library interface that enables them to access the various cryptographic algorithm functions supplied by the module.

The module uses Secure multi-party computation to implement NIST-validated cryptographic algorithms.

2.2.1 Hardware, Software and Firmware Components

The module is a software module that resides on the hardware of a general-purpose computer (see Figure 3). For the purposes of FIPS 140-2 testing, the module is evaluated running on the operational environments defined in section 2.6.

The module is packaged as a number of distinct binary images:

OS Family	Filename(s)	SHA-256 Hash
Windows	ekmengine_jni.dll	011a6e24cc9625ba72d03fffca8ddfd3b4df1d85808b97ada039493d6702161b
	libeay32.dll	1bb766e875b89dedfe91e399778968f4b34dd19ace83306a038d4f4af4db018e
	ssleay32.dll	9bd3f667dcc89a5e6e1e8f3d6706c9b5c0bf5ef679a98ebe172dbf20e022fcad

Figure 2 Module Binary Images

2.2.2 Cryptographic Boundary

The physical boundary of the module encompasses multiple general-purpose computers (GPCs) on which it is installed (see Figure 3). Each instance is a software module running in a well-defined operational environment on a general-purpose computer. The processor of this platform executes all software. All software components of the module are persistently stored within the device and, while executing, are stored in the device local RAM.

Within the cryptographic boundary of the module is an unmodified copy of the OpenSSL FIPS Object Module version 2.0.16.

The cryptographic boundary of the module is shown in Figure 4. The only software within the logical boundary of the cryptographic boundary is listed in Figure 2.

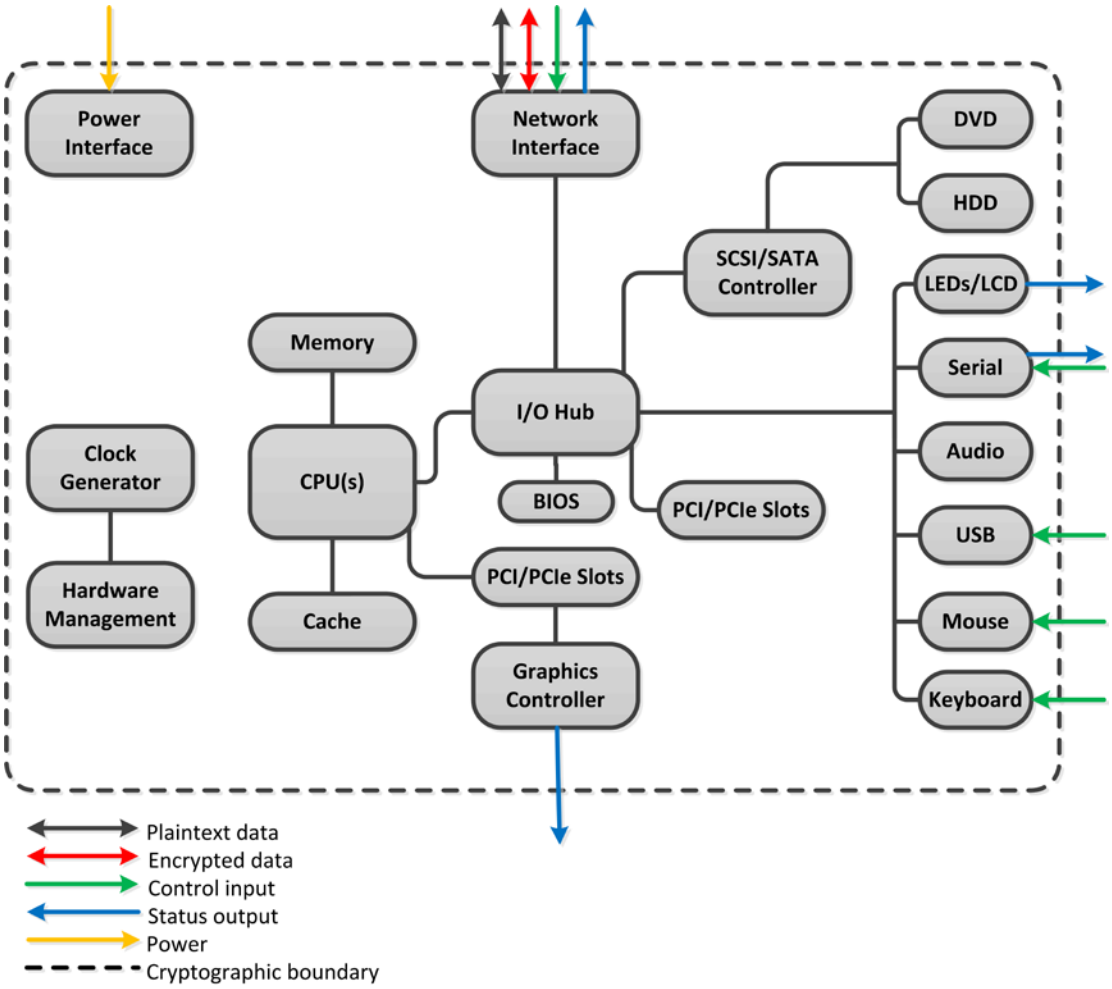

Figure 3 General-purpose Computer Hardware Block Diagram

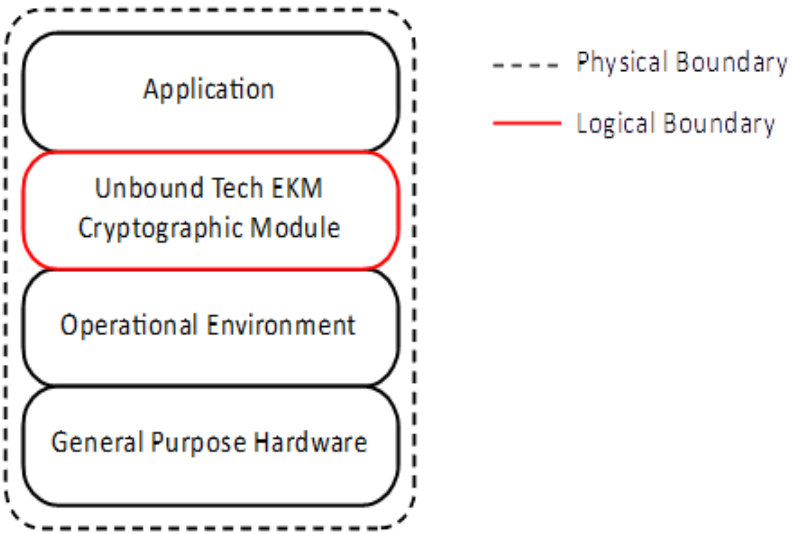

Figure 4 Logical Diagram of the Cryptographic Boundary

The EKM Cryptographic Module is constructed from three (3) different components: Entry, Pair and Auxiliary. All three (3) components take part in executing the cryptographic module and executing crypto operations. Together they form the logical EKM Cryptographic Module. The different modules communicate over secure channels. The generic structure of the different components is shown in Figure 5.

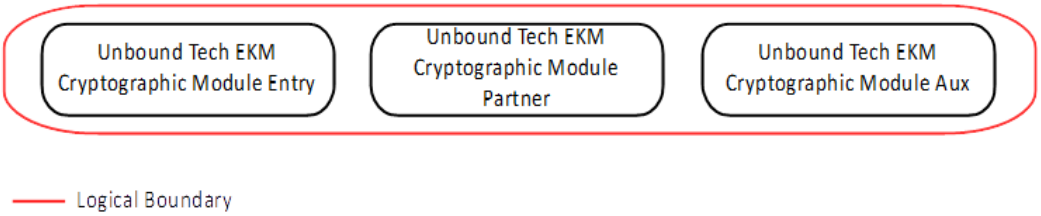

Figure 5 Logical Diagram of the EKM Cryptographic Module

The three different components of the cryptographic module can be configured to run on different physical and logical boundaries but is always executing the same code on each of the three (3) components and communicating in the same way over secured channels. The certification should test and certify all configurations as listed below:

1. Each logical component of the Cryptographic Module is installed on a different machine (Figure 6).
2. Each logical component of the Cryptographic Module is installed on a different virtual machine running in a single hypervisor on a single physical machine (Figure 7).
3. Each logical component of the Cryptographic Module is installed on a different Docker container running on a single machine (Figure 8).
4. Each logical component of the Cryptographic Module is part of a different process running on the same machine (Figure 9).
5. All logical components of the Cryptographic Module are part of a single process (Figure 10).

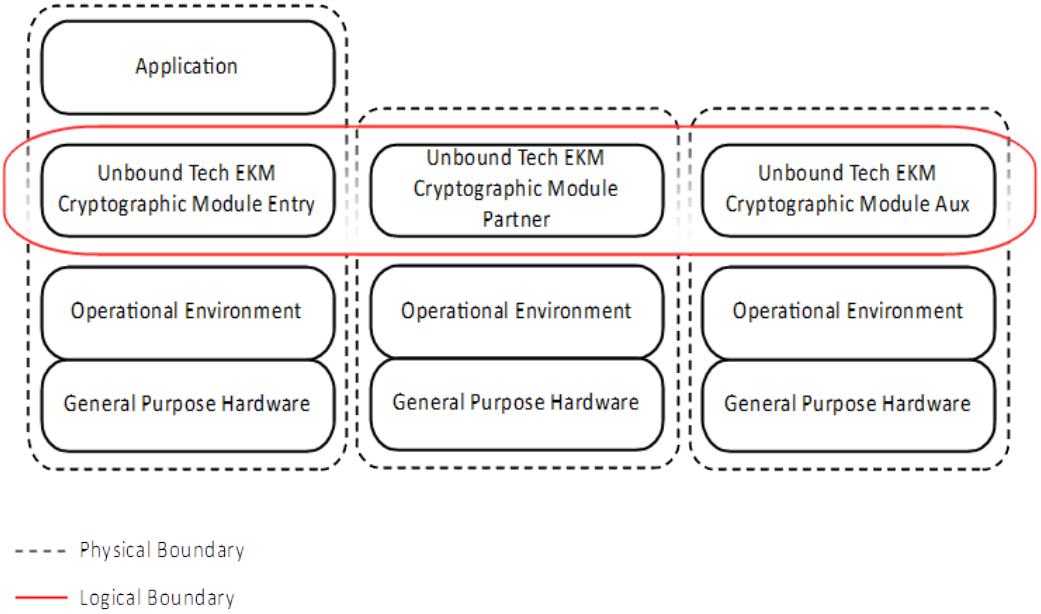

Figure 6 Logical Diagram of the EKM Cryptographic Module Running on Different Machines

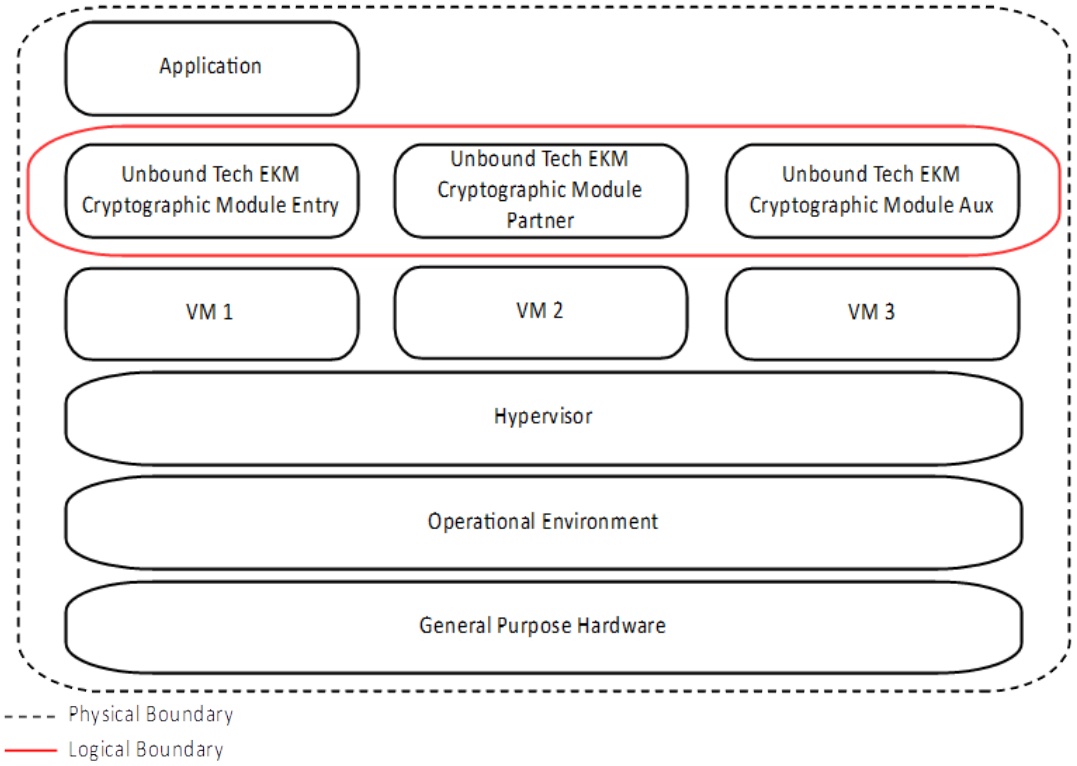

Figure 7 Logical Diagram of the EKM Cryptographic Module Running on Different Virtual Machines in the Same Hypervisor

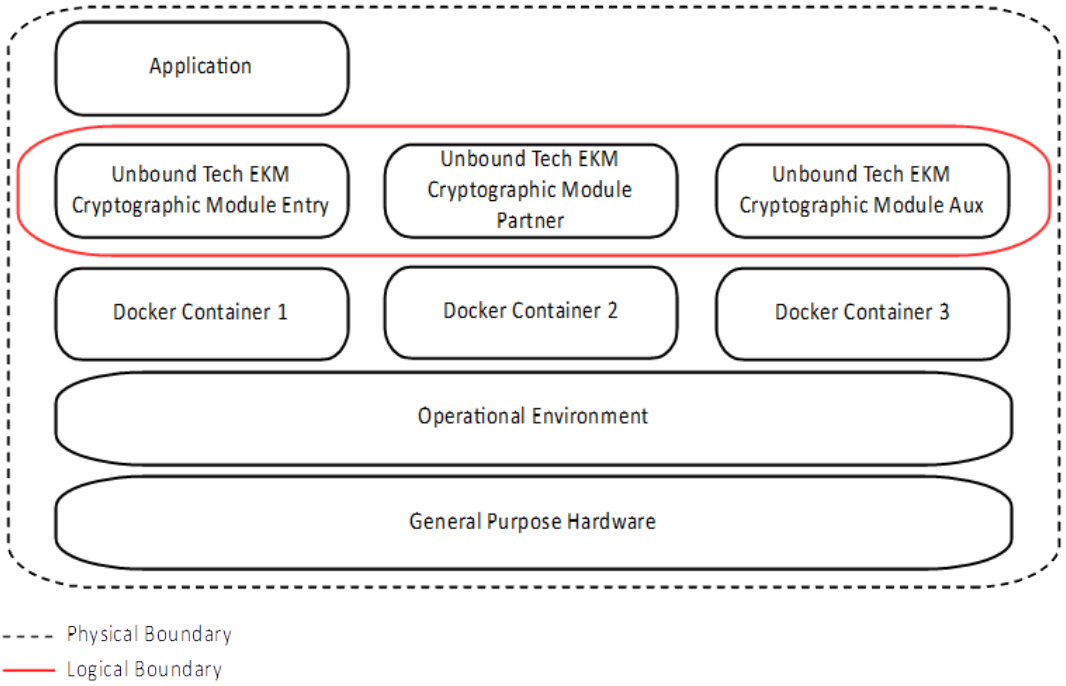

Figure 8 Logical Diagram of the EKM Cryptographic Module Running on Different Docker Containers

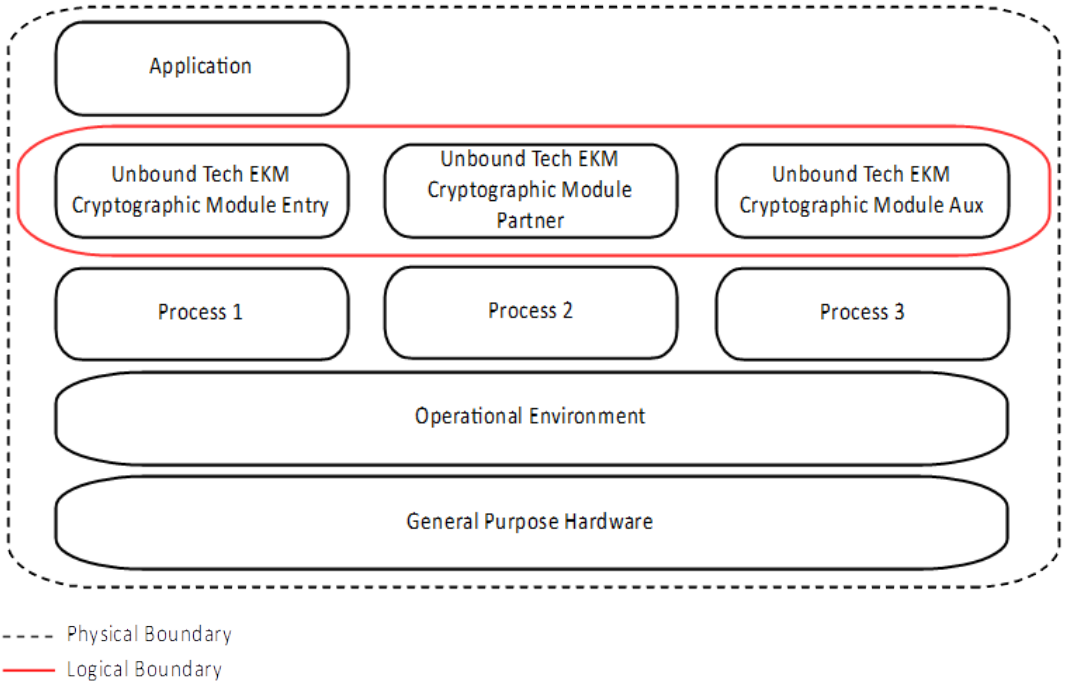

Figure 9 Logical Diagram of the EKM Cryptographic Module Running on Different Processes

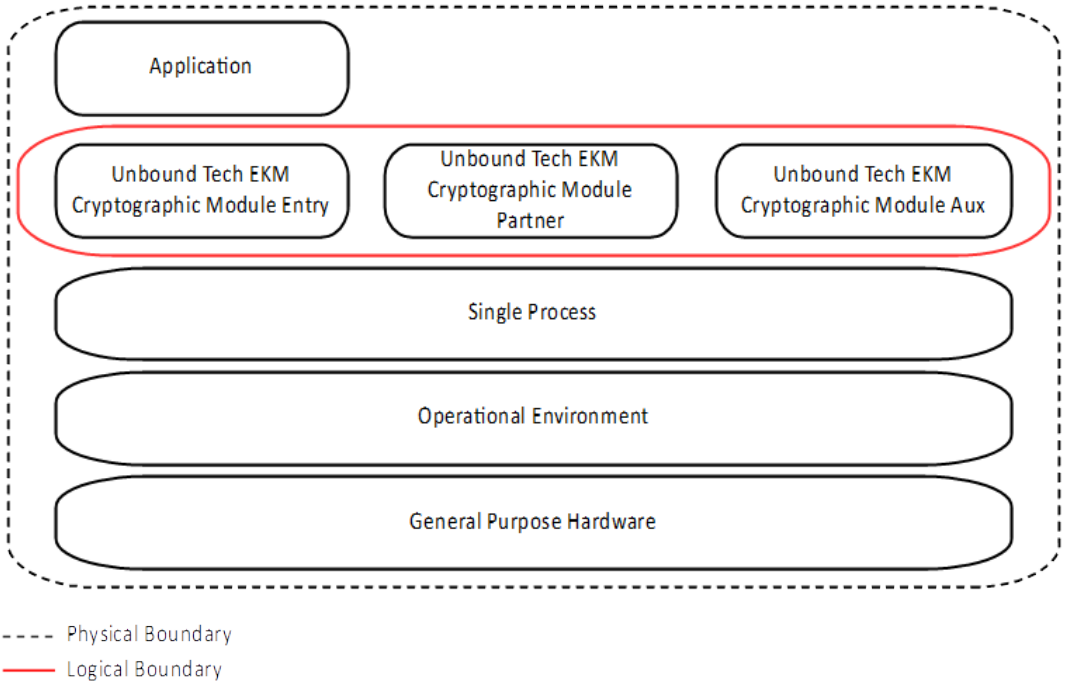

Figure 10 Logical Diagram of the EKM Cryptographic Module Running on a Single Process

2.2.3 Scope of Evaluation

The cryptographic module meets the overall requirements applicable to Level 2 security of FIPS 140-2, with Design Assurance at Level 3.

Security Requirements Section	Level
Cryptographic Module Specification	2
Module Ports and Interfaces	2
Roles, Services and Authentication	2
Finite State Model	2
Physical Security	N/A
Operational Environment	2
Cryptographic Key Management	2
EMI/EMC	2
Self-Tests	2
Design Assurance	3
Mitigation of Other Attacks	2

Figure 11 Security Level Specification per FIPS 140-2 Section

2.2.4 Cryptographic Algorithms

2.2.4.1 Approved Algorithms

The following tables provide details of the approved algorithms that are included within the module. The information is separated into two (2) tables to differentiate between algorithms that are contained within the EKM component and those that are contained within the OpenSSL component. (Note, the EKM DRBG is not distributed, though its seeding process is). The module supports both an Approved mode and non-Approved mode of operation. The module alternates service by service between approved and non-approved modes of operations. The module is considered in the Approved mode when services in Figure 16 are called.

Items in {curly brackets} are used only during power-on self-tests.

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves, Moduli, etc.	Use
5444	AES	FIPS 197, SP 800-38A SP 800-38B SP 800-38C SP 800-38D	ECB, CBC, CFB128, OFB, CCM, GCM*, CTR, CMAC	128, 192, 256	Data Encryption/Decryption, Message Authentication
5444	AES	FIPS 197, SP 800-38E	XTS*	128, 256	Data Encryption/Decryption
5444	AES	FIPS 197, SP 800-38F	KW[1]	128, 192, 256	Key Wrapping/Unwrapping
Vendor Affirmed	CKG	SP 800-133	§6.1, §6.2 (Asymmetric) §7.1 (Direct Symmetric)	Varies (128 to 4096)	Key generation for other algorithms performed by this module
2126	DRBG	SP800-90A	CTR-DRBG (with DF)	AES-256	Deterministic Random Bit Generation
1448	ECDSA	FIPS 186-4	N/A	P-256, P-384, P-521	Key Pair Generation (Appendix B.4.1)
1888	ECDSA (CVL)	FIPS 186-4	N/A	P-256, P-384, P-521	Digital Signature Generation Component
3601	HMAC	FIPS 198-1	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	128-1024, 128-2048, 128-3072, 128-4096 (λ = hash size)	Message Authentication
1887	KAS EC DH (CVL: primitive only)	SP 800-56Ar1	ECC	P-256, P-384, P-521	Key Agreement (component)
Vendor Affirmed	KAS EC DH	SP 800-56Ar2	ECC SP800-135 KDF	P-256, P-384, P-521	Key Agreement

[1] AES Key Wrap (KW) mode

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves, Moduli, etc.	Use
5444	KTS	SP 800-38F	AES-KW	128, 192, 256	Key Wrapping, Key Unwrapping
Vendor Affirmed	KTS-RSA	SP 800-56B	OAEP	2048, 3072, 4096	Key Wrapping, Key Unwrapping
2919	RSA	FIPS 186-4	X9.31 (Probable Primes, Appendix B.3.3)	2048, 3072, 4096	Key Pair Generation
2919	RSA	FIPS 186-4	PKCS1 v1.5	2048, 3072, 4096	Digital Signature Generation
2919	RSA	FIPS 186-4	PSS	2048, 3072, 4096	Digital Signature Generation
1889	RSA-DP (CVL)	SP 800 56B	N/A	2048	Decryption Primitive

Figure 12 Approved Algorithms (EKM)

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves, Moduli, etc.	Use
5443	AES	FIPS 197, SP 800-38A, SP 800-38D	ECB, CBC, CFB128, OFB, GCM*	128, 192, 256	Data Encryption/Decryption, Message Authentication
1885	CVL TLS v1.2 KDF	SP 800-135 rev1	TLS v1.2	N/A	Key Derivation
1447	ECDSA	FIPS 186-4	N/A	{P-224}, P-256, P-384, P-521	Key Pair Generation (Appendix B.4.1), Digital Signature Generation, Digital Signature Verification
1886	ECDSA (CVL)	FIPS 186-4	N/A	{P-224}, P-256, P-384, P-521	Digital Signature Generation, Component
3600	HMAC	FIPS 198-1	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	128-1024, 128-2048, 128-3072, 128-4096	Message Authentication
1884	KAS EC DH (CVL: All except KDF)	SP 800-56A	ECC	EB (P-224) EC (P-256), ED (P-384), EE (P-521)	Key Agreement
2918	RSA	FIPS 186-4	PKCS v1.5	1024[2], 2048, 3072, 4096[3]	Digital Signature Generation, Digital Signature Verification
2918	RSA	FIPS 186-4	PSS	1024[4], 2048, 3072, 4096[5]	Digital Signature Generation, Digital Signature Verification

[2] 1024 – legacy verification only
[3] 4096 – generation only
[4] 1024 – legacy verification only
[5] 4096 – generation only

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves, Moduli, etc.	Use
4362	SHS	FIPS 180-4	SHA-1[6], SHA-256, SHA-384, SHA-512	N/A	Message Digest

Figure 13 Approved Algorithms (OpenSSL)

* The module ensures XTS key pairs are not identical. AES XTS is for use in storage applications only. Refer to Section 3.2 for notes on GCM IV behavior.

2.2.4.2 Non-Approved Algorithms Allowed in Approved Mode

- NDRNG (seeds DRBG with 384,000 bits of Entropy Input + Nonce)
- EC Diffie-Hellman (key agreement; key establishment methodology provides 128 bits of encryption strength)
- RSA unwrap, non-compliant to SP800-56B (allowed by IG D.8; PKCS 1.5 padding). Legacy use only.

2.2.4.3 Non-Approved Algorithms

- Password Protection (proprietary)
- Triple-DES
- AES GMAC (non-compliant)

2.2.4.4 TLS Supported Cipher Suites

The module supports the following cipher suites for TLS v1.2 (older TLS versions are not supported):

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA384

2.2.5 Components Excluded from the Security Requirements of the Standard

There are no components excluded from the security requirements of the standard.

[6] Disallowed for digital signature generation , except where specifically allowed by NIST protocol-specific guidance; allowed for legacy use for digital signature verification; acceptable for non-digital signature uses.

2.3 Physical Ports and Logical Interfaces

The module is classified as a multi-chip standalone module for FIPS 140-2 purposes. The module's physical boundary encompasses the general-purpose computer on which it is installed as well as the other instances it is bound to (entry, pair, auxiliary).

The module provides its logical interfaces via Application Programming Interface (API) calls. This logical interface exposes services (described in section 2.4.2) that the User and operating system utilize directly.

The logical interfaces provided by the module are mapped onto the FIPS 140-2 logical interfaces: data input, data output, control input, and status output as follows:

FIPS 140-2 Logical Interface	Module Mapping
Data Input	Parameters passed to the module via API calls
Data Output	Data returned from the module via API calls
Control Input	API Calls and/or parameters passed to API calls
Status Output	Information received in response to API calls
Power Interface	There is no separate power or maintenance access interface beyond the power interface provided by the GPC itself

Figure 14 Module Interfaces

2.4 Roles, Services and Authentication

2.4.1 Roles

The Cryptographic Module implements both a Crypto Officer role and a User role. The module uses certificate-based authentication to allow an operator to assume a role. The authentication process requires the operator to sign a challenge provided by the module that the module then authenticates the response. Section 2.4.2 summarizes the services available to each role.

Role	Description
Crypto Officer (CO)	The administrator of the module having full configuration and key management privileges.
User	General User of the module

Figure 15 Roles

2.4.2 Services

Most of the services provided by the module are provided via access to API calls using interfaces exposed by the module.

However, some of the services, such as power-up module integrity testing are performed automatically and so have no function API, but do provide status output. Services listing N/A as their Role are unauthenticated.

For services involving asymmetric cryptography, the module only provides services relating to the private key.

	Service	Description	Approved Mode	Role(s)	Keys and CSPs	CAVP Cert #	RWE
1	Import RSA Key	Creates an RSA key object from the imported data.	Yes	CO, User	RSA key share	#2919	W
2	Import EC DH Key	Creates an EC DH key object from the imported data.	Yes	CO, User	EC DH key share	#1888	W
3	Import ECDSA Key	Creates an ECDSA key object from the imported data.	Yes	CO, User	ECDSA key share	#1448	W
4	Import AES Key	Creates an AES key object from the imported data.	Yes	CO, User	AES key share	#5444	W
5	Import HMAC Key	Creates a new HMAC key from the imported data.	Yes	CO, User	HMAC key share	#3601	W
6	Generate RSA Key	Randomly generates a new RSA key.	Yes	CO, User	RSA key share DRBG State DRBG Entropy Input	#2919	W
7	Generate EC DH Key	Randomly generates a new EC DH key.	Yes	CO, User	EC DH key share DRBG State DRBG Entropy Input	#1888	W
8	Generate ECDSA Key	Randomly generates a new ECDSA key.	Yes	CO, User	ECDSA key share DRBG State DRBG Entropy Input	#1448	W
9	Generate AES Key	Randomly generates a new AES key.	Yes	CO, User	AES key share DRBG State DRBG Entropy Input	#5444	W
10	Generate HMAC Key	Randomly generates a new HMAC key.	Yes	CO, User	HMAC key share DRBG State DRBG Entropy Input	#3601	W
11	Export a Key	Exports a key in full from the module[7]	Yes	CO, User	Any secret or private key	#2919 #1888 #1448 #5444 #3601	R
12	Sign PKCS1	Signs data using PKCS #1 with MPC	Yes	CO, User	RSA key share	#2919	E
13	Sign PSS	Signs data using RSA PSS with MPC	Yes	CO, User	RSA key share	#2919	E
14	Decrypt RSA Raw	Performs a raw RSA decrypt using the private key using MPC	Yes	CO, User	RSA key share	#1889	E

[7] Only possible if the key was defined as "exportable" when generated. Most of the keys are generated with this flag set to false.

	Service	Description	Approved Mode	Role(s)	Keys and CSPs	CAVP Cert #	RWE
15	Decrypt PKCS1	Decrypts data using PKCS #1 with MPC	Yes	CO, User	RSA key share	#2919	E
16	Decrypt OAEP	Decrypts data using RSA OAEP with MPC	Yes	CO, User	RSA key share	#2919	E
17	ECDSA	Signs hashed data using ECDSA with MPC	Yes	CO, User	ECDSA key share	#1448	E
18	EC DH	EC DH key agreement computation using MPC	Yes	CO, User	EC DH key share	#1888	E
19	Wrap AES NIST	SP 800-38F key wrapping	Yes	CO, User	AES key share	#5444	E
20	Get key informati on	Return key information on a key share	Yes	CO, User	All keys	N/A	R
21	Load key	Loads key share from binary input to module.	Yes	CO, User	RSA, ECDSA, EC DH, AES, HMAC key share	#2919 #1888 #1448 #5444 #3601	W
22	Delete key	This function deletes a key-share from the local module only. It does not affect any remote servers	Yes	CO, User	RSA, ECDSA, EC DH, AES, HMAC key share	#2919 #1888 #1448 #5444 #3601	W
23	Refresh key	Performs a refresh of key-shares in the module	Yes	CO, User	RSA, ECDSA, EC DH, AES, HMAC key share	#2919 #1888 #1448 #5444 #3601	W
24	Read Key	Outputs the key share from module as binary data	Yes	CO, User	RSA, ECDSA, EC DH, AES, HMAC key share	#2919 #1888 #1448 #5444 #3601	R
25	Get public key	Returns the public key corresponding to the identified private key (RSA, EC DH or ECDSA)	Yes	CO, User	All public keys	#2919 #1888 #1448	R
26	AES ECB	AES ECB encryption and decryption	Yes	CO, User	AES key share	#5444	E
27	AES CBC	AES CBC encryption and decryption using MPC	Yes	CO, User	AES key share	#5444	E
28	AES OFB	AES OFB encryption and decryption using MPC	Yes	CO, User	AES key share	#5444	E
29	AES CFB	AES OFB encryption and decryption using MPC	Yes	CO, User	AES key share	#5444	E
30	AES CCM	AES CCM encryption and decryption using MPC	Yes	CO, User	AES key share	#5444	E
31	AES CTR	AES CTR using MPC	Yes	CO, User	AES key share	#5444	E
32	AES XTS	AES XTS using MPC	Yes	CO, User	AES key share	#5444	E
33	AES GCM	AES GCM using MPC	Yes	CO, User	AES key share	#5444	E

	Service	Description	Approved Mode	Role(s)	Keys and CSPs	CAVP Cert #	RWE
36	HMAC	HMAC using MPC	Yes	CO, User	HMAC key share	#3601	E
37	Show status	Returns self-test results, FIPS mode status and module version number	N/A	N/A[8]	N/A	N/A	N/A
38	Self-tests	Performs the power-up self-tests	N/A	N/A	Module Integrity Public Key	N/A	N/A
39	Zeroize keys	Zeroizes all secret and private keys in the module	N/A	N/A	All keys and CSPs	N/A	W
40	Register nodes[9]	Allows nodes to be registered to form a coordinated module	N/A	N/A	N/A	N/A	N/A
41	Install OpenSSL certificate [10]	Installs the OpenSSL certificate used for inter-node authentication	N/A	CO	TLS Peer Public Key	#1447 #2918	W
42	Initialize module[11]	Starts up the module (initiates power-up self-tests, establishes TLS tunnels for distributed operation)	N/A	CO	TLS keys and CSPs DRBG State DRBG Entropy Input Operator authentication public key	#5443 #1884 #1885 #1886 #1887 #3600 #4362	RWE
43	Stop module	Stops the module and zeroizes keys	N/A	N/A	All keys and CSPs	N/A	W

Figure 16 FIPS Approved Services

2.4.3 Authentication

The operator passes a signed ticket into the library via a logon API for authentication. If this authentication is successful, it passes back a handle that the operator can use to access module services. Available signature algorithms for operator authentication are:

- RSA-2048 w/SHA-256 (PKCS1 padding)
- ECDSA P-256, P-384, or P-521 w/SHA-256

[8] Note that if the Role is marked N/A this indicates that the service is unauthenticated.

[9] Each server (Entry, Partner and Aux) registers itself with its own hostname (or IP), entry point and partner will also register their buddy (partner and entry respectively) and both will register the Aux. Aux only registers itself.

[10] Installs the peer self-signed certificate and the local server private key so all parties can establish trust during the TLS tunnel establishment process.

[11] Starts the library and runs the FIPS 140-2 power-up self-tests. This service runs automatically when the module is loaded.

A signed ticket cannot be forged without breaking the Approved signature algorithm (RSA or ECDSA). The weakest of these algorithms is RSA-2048 with 112 bits of strength, so the probability of false authentication for any given attempt is 1 in 2^112, which is less than 1 in 1,000,000.

The probability of false authentication over a given time period depends on the computational power of the adversary but in any reasonable scenario (i.e. one where an attacker cannot break an Approved algorithm) will be less than 2^80 attempts per second. This corresponds to a 60 in 2^32 probability of success, which is less than 1 in 100,000.

2.5 Physical Security

The Cryptographic Module is a software-only cryptographic module and therefore the physical security requirements of FIPS 140-2 do not apply.

2.6 Operational Environment

The Cryptographic Module has been tested on and found to be conformant with the requirements of FIPS 140-2 overall Level 2. The module is installed and run on a NIAP certified Common Criteria operating system and configured to run in a Common Criteria compliant mode of operation (https://www.niap-ccevs.org/Product/CompliantCC.cfm?CCID=2019.1018). The module runs on the following GPC platforms:

Operating System	Platform	CPU	AES-NI
Windows Server 2016	Gigabyte GA-6LISL[12] - Figure 7	Intel Core i3	X
Windows Server 2016	Gigabyte GA-6LISL - Figure 7	Intel Core i3	

Figure 17 Certified Operational Environments

As shown above, testing has been performed both with and without AES-NI, as applicable.

While not tested as part of this validation, the module is capable of running on other Windows and Linux based platforms. Per IG G.5, the vendor affirms compliance so long as the operating system is on the NIAP PCL per FIPS 140-2 Annex B. Currently this is the case for the following operating systems:

- Windows Server 2016 on ESXi 6.5
- Oracle Linux 7.3 OSPP
- Windows 10
- Windows 10 on ESXi 6.5

When run on Linux the module is recompiled and packaged as the following distinct binary images:

OS Family	Filename(s)	SHA-256 Hash
Linux	libekmengine_jni.so	108c49bafa8870dca2ebbba6a509196c937926f56466a60350a1cbaa882753d8
	libcrypto.so.1.0.0	c0ff6afe1ff9a6fac4fc98cfc08019d48c857ed2765aea7915c1c969a3a92bcd
	libssl.so.1.0.0	

[12] The Gigabyte GA-6LISL is just a motherboard. This is the platform used for testing the Cryptographic Module.

OS Family	Filename(s)	SHA-256 Hash
		069582f338a3d5f67502e5e79d3846decad59a22f34d54d0757bde2e38c36019

Figure 18 Linux Module Binary Images

The cryptographic module runs in the thread context of the calling application. This provides it with protection from all other processes, preventing access to all keys, intermediate key generation values, and other CSPs.

The task scheduler and architecture of the operating system maintain the integrity of the cryptographic module.

The module supports a single operator and does not support multiple concurrent operators. The execute only attribute is set on the module binary file in order to prevent unauthorized read or write operations.

Both of the supported Cryptographic Module roles described in section 2.4.1 map to the User Role specified for the Common Criteria Operating System.

The Cryptographic Module uses Syslog to record all access to services.

2.7 Cryptographic Key Management

2.7.1 Random Number Generators

The module supports an SP800-90A CTR DRBG for the generation of all internally-generated keys. The DRBG seeding process depends on the operational environment but has been verified to provide (for all tested OEs) a minimum of 755 bits of entropy to seed the DRBG.

2.7.2 Key Generation & Establishment

EC DH Key Agreement provides a maximum of 256 bits of security strength. RSA key unencapsulation provides between 112 bits of security strength (from 2048-bit RSA keys) and 128+ bits of security strength (from 4096-bit RSA keys).

AES key wrapping provides the security strength of the strength of the lesser of the wrapping key and the wrapped key.

Private and secret keys are generated by the module using the EKM CTR-DRBG and entropy from the calling application.

The output from this DRBG seeded by the supplied entropy is used as the basis for key generation. AES keys are created directly from the DRBG output, whilst this output is used in the creation of private keys according to the requirements of FIPS 186-4.

2.7.3 Key Tables

The TLS keys depend on the cipher suite (Signing key is RSA or ECDSA, exchange key is always EC DH). A self-signed certificate is used. Ephemeral Diffie-Hellman is used to give forward secrecy. The default cipher suite used by the module is TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384. Each node is both a TLS server and client when it connects to another node, so each node-node connection requires two TLS tunnels.

Name	Purpose	Generation	Storage, I/O	Destruction
TLS Private Key. (RSA 2048/3072/4096-bit, ECDSA P-256, P-384, P-521)	Used to establish TLS tunnel for authentication of communication between distributed nodes	Module generates keys internally using supplied random data.	Stored in RAM. I/O via API*.	Library shutdown.
TLS ECDHE Key. (P-256)	Used during TLS handshake (creates TLS Master Secret)	Internally generated.	Stored in RAM.	TLS handshake completes.
TLS Master Secret. (384 bits)	Used by TLS KDF to generate TLS Session Keys	ECDHE exchange (KAS-ECC) and TLS KDF.	Stored in RAM.	TLS session terminates.
TLS Session Keys. (128, 256-bit AES GCM)	Used for TLS authenticated encryption.	Derived from TLS KDF.	Stored in RAM.	TLS session terminates.
DRBG Entropy Input (Size varies)	Seeds DRBG.	Internally generated.	Stored in RAM.	DRBG seeding completes.
DRBG State (128-bit V, 256-bit Key)	Used by DRBG.	Internally generated.	Stored in RAM.	Library shutdown.
Key shares. (AES, RSA, ECC, HMAC, ECDSA)	Distributed key components used for operations requiring secret or private keys. Key shares are the same size as the whole key and reveal nothing about the whole key in isolation.	Internally generated.	Stored in RAM. I/O via API*.	Key zeroization service.

Figure 19 Module CSPs

Name	Purpose	Generation	Storage, I/O	Destruction
TLS Public Key (RSA 2048/3072/4096-bit, ECDSA P-256, P-384, P-521)	Used to establish TLS tunnel for authentication of communication between distributed nodes	Module generates RSA keys internally using supplied random data.	Stored in RAM. I/O via API*. Output during TLS handshake.	Library shutdown
TLS Peer Public Key (RSA 2048/3072/4096-bit, ECDSA P-256, P-384, P-521)	Used to establish TLS tunnel for authentication of communication between distributed nodes	Externally generated.	Provided by TLS peer. Stored in RAM.	Library shutdown
TLS ECDHE Public Keys (P-256)	Used during TLS handshake (creates TLS Master Secret)	Internally generated	Received from client and provided to client during TLS handshake. Stored in RAM.	TLS handshake completes.
Public key shares. (RSA, ECC, HMAC, ECDSA)	Distributed key components used for operations requiring secret or private keys. Key shares are the same size as the whole key and reveal nothing about the whole key in isolation.	Internally generated.	Stored in RAM, I/O via API*.	Key zeroization service
Module integrity public key (RSA 2048-bit)	Used during software integrity test	Externally generated	Stored on disk, and in RAM	N/A
Operator authentication public key (RSA/ECC)	Use to verify the signature on the ticket presented during operator authentication	Externally generated	Stored on disk, and in RAM. Input via API*.	N/A

Figure 20 Module Public Keys

* These methods use a GPC internal path, which is N/A as per FIPS IG 7.7.

2.7.4 Key Destruction

Key zeroization can be achieved using the Zeroize keys service to actively remove all plaintext secret and private keys from the module.

2.7.5 Access to Key Material

See Figure 16, column 8 for details.

2.8 Self-Tests

In normal operation, the module uses MPC techniques to spread the load of cryptographic operation between several nodes. However, for self-testing, the module is able to perform all of the steps of each algorithm within the boundary of the module.

The module implements both power-up and conditional self-tests as required by FIPS 140-2.

The following two sections outline the tests that are performed.

2.8.1 Power-up Self-tests

At startup the module executes the Power-Up Self-Tests with no further inputs or actions by the operator.

The module implements the following power-up self-tests. The module inhibits all data output while it is operating in the Self-Test state.

Object	Test
AES	AES-128 ECB encrypt known answer test AES-128 ECB decrypt known answer test
RSA	Signature generation known answer test (2048-bit) Signature verification known answer test (2048-bit) Decryption known answer test (2048-bit)
ECDSA	Signature generation known answer test (P.256 curve)
EC DH	Primitive "Z" computation known answer test (P.256 curve)
HMAC	HMAC-SHA-256 known answer test
AES CMAC	AES-128 CMAC known answer test
AES key wrap	SP 800-38F key wrapping known answer test (wrap and unwrap)
Module integrity	SHA-256 hash of the code has a 2048-bit RSA signature used for module integrity test
DRBG	SP 800-90 CTR-DRBG section 11.3 Health tests DRBG KAT

Figure 21 EKM Power-up Self-tests

Object	Test
AES	AES-128 ECB encrypt known answer test AES-128 ECB decrypt known answer test AES-256 GCM encrypt known answer test AES-256 GCM decrypt known answer test AES-128 XTS encrypt known answer test AES-128 XTS decrypt known answer test AES-256 XTS encrypt known answer test AES-256 XTS decrypt known answer test
RSA	Signature generation known answer test (2048-bit)

Object	Test
	Signature verification known answer test (2048-bit) Decryption known answer test (2048-bit)
ECDSA	Signature generation known answer test (P.224 curve) Signature verification known answer test (P.224 curve)
EC DH	Primitive "Z" computation known answer test (P.224 curve)
SHS	SHA-1 known answer test SHA-256 known answer test SHA-384 known answer test SHA-512 known answer test
HMAC	HMAC-SHA-1 known answer test HMAC-SHA-256 known answer test HMAC-SHA-384 known answer test HMAC-SHA-512 known answer test

Figure 22 OpenSSL Power-up Self-tests

If any of the power-up known answer tests fail, the module exits with an error. While in the error state the module inhibits all data output and all cryptographic operations are prohibited. The operator may restart the module to re-run the power up self-tests.

In addition, the module performs the FIPS-mode OpenSSL power-up self-tests.

2.8.2 Conditional Self-tests

The module implements the following conditional self-tests:

Event	Test	Consequence of failure
RSA key pair is generated	RSA pair-wise consistency test (two part: sign-verify and encrypt-decrypt)	Generated key pair is discarded
ECDSA key pair is generated	ECDSA pair-wise consistency test	Generated key pair is discarded
DRBG is invoked	DRBG continuous RNG test	The module returns an error
DRBG is seeded	NDRNG continuous RNG test	The module returns an error

Figure 23 Conditional Self-tests

2.9 Design Assurance

Unbound Tech employs industry standard best practices in the design, development, production and maintenance of all of its products, including the FIPS 140-2 module.

This includes the use of an industry standard configuration management system that is operated in accordance with the requirements of FIPS 140-2, such that each configuration item that forms part of the module is stored with a label corresponding to the version of the module and that the module and all of its associated documentation can be regenerated from the configuration management system with reference to the relevant version number.

Design documentation for the module is maintained to provide clear and consistent information within the document hierarchy to enable transparent traceability between corresponding areas

throughout the document hierarchy, for instance, between elements of this Cryptographic Module Security Policy (CMSP) and the design documentation.

Guidance appropriate to an operator's Role is provided with the module and provides all of the necessary assistance to enable the secure operation of the module by an operator, including the Approved security functions of the module.

Delivery of the Cryptographic Module to customers from the vendor is via secure download. The module software downloaded can be verified using SHA-256 hash values that are downloaded separately.

2.10 Mitigation of Other Attacks

The module provides key security over and above that required by FIPS 140-2.

The use of MPC techniques within the module ensures that knowledge of a key share provides no information about the logical key that it is a part of and that compromising a single module gives an attacker no knowledge about the secret and private keys used by that module.

3 Secure Operation

The module is delivered to an end-user as part of a larger system or product. There are no configuration or installation options required for the module.

To ensure secure delivery of the module, a hash value of the certified module is available from Unbound Tech upon request and this can be verified using a third-party hash tool.

3.1 Security Rules & Guidance

This section documents the security rules for the secure operation of the cryptographic module to implement the security requirements of FIPS 140-2.

1. The module provides two distinct operator roles: User and Cryptographic Officer.
2. The module provides role-based authentication.
3. The module clears previous authentications on power cycle.
4. An operator does not have access to any cryptographic services prior to assuming an authorized role.
5. The module allows the operator to initiate power-up self-tests by resetting the module.
6. Power up self-tests do not require any operator action.
7. Data output are inhibited during key generation, self-tests, zeroization, and error states.
8. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
9. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
10. The module does not support concurrent operators.
11. The module does not support a maintenance interface or role.
12. The module does not support manual key entry.

13. The module does not have any proprietary external input/output devices used for entry/output of data.
14. The module does not enter or output plaintext CSPs.
15. The module does not output intermediate key values.
16. The following steps are necessary in order to connect the nodes:
    - Register servers (entry, partner and auxiliary nodes)
    - Set (User and Crypto Officer) Authentication certificates
    - Set OpenSSL Certificates
    - Start Module (for each server: entry, partner and auxiliary nodes)

3.2 Notes on GCM

Depending on the implementation, GCM is handled in two separate manners:

3.2.1 OpenSSL

The OpenSSL implementation of GCM is used solely within the context of TLS v1.2 (IG A.5 option 1a). This implementation of TLS v1.2 occurs entirely within the module and is compliant with the appropriate RFCs (5116, 5288, and 5289) as per IG A.5 Option 1. It has been verified during operational testing to behave correctly. Overflow of the `nonce_explicit` would require 2^64 (~18.4 billion billion) messages within the same TLS session, so this is expected to never occur. Power failure (or any other form of module uninstantiation, planned or otherwise) terminates all TLS sessions, and by extension the associated GCM keys, forcing operators to re-establish them.

3.2.2 EKM

The EKM implementation of GCM is used by the operator for arbitrary operations (IG A.5 option 3). The module internally constructs IVs using operator-provided initial IVs and the message sequence number (increments `nonce_explicit`). Similar to the OpenSSL implementation, 2^64 invocations would be required for the `nonce_explicit` to overflow. Power failure (or any other form of module uninstantiation, planned or otherwise) causes loss of all GCM keys and IVs, such that GCM operation does not resume unless module operators re-establish keys and IVs.