Stryker Corporation

Stryker Badge Cryptographic Module

Version: 3.4.0

FIPS 140-3 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.0

Prepared for:

Stryker Corporation
3030 Orchard Parkway
San Jose, CA 95134
United States of America

Phone: +1 800 331 6356
www.stryker.com

Prepared by:

Corsec Security, Inc.
12600 Fair Lakes Circle, Suite 210
Fairfax, VA 22033
United States of America

Phone: +1 703 267 6050
www.corsec.com

Table of Contents

List of Tables

List of Figures

1. General

1.1 Overview

This is a non-proprietary Cryptographic Module Security Policy for the Stryker Badge Cryptographic Module (firmware version: 3.4.0) from Stryker Corporation (Stryker). This Security Policy describes how the Stryker Badge Cryptographic Module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-3, which details the U.S. and Canadian government requirements for cryptographic modules. More information about the FIPS 140-3 standard and validation program is available on the Cryptographic Module Validation Program (CMVP) website, which is maintained by the National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS).

This document also describes how to run the module in a secure Approved mode of operation. This policy was prepared as part of the Level 1 FIPS 140-3 validation of the module. The Stryker Badge Cryptographic Module is referred to in this document as "Stryker Badge Crypto Module" or "module".

1.1.1 References

This document deals only with operations and capabilities of the module in the technical terms of a FIPS 140-3 cryptographic module security policy. More information is available on the module from the following sources:

- The Stryker website (www.stryker.com) contains information on the full line of products from Stryker.
- The search page on the CMVP website (https://csrc.nist.gov/Projects/cryptographic-module-validation-program/Validated-Modules/Search) can be used to locate and obtain vendor contact information for technical or sales-related questions about the module.

1.1.2 Document Organization

ISO/IEC 19790 Annex B uses the same section naming convention as *ISO/IEC 19790* section 7 - Security requirements. For example, Annex B section B.2.1 is named "General" and B.2.2 is named "Cryptographic module specification," which is the same as *ISO/IEC 19790* section 7.1 and section 7.2, respectively. Therefore, the format of this Security Policy is presented in the same order as indicated in Annex B, starting with "General" and ending with "Mitigation of other attacks." If sections are not applicable, they have been marked as such in this document.

1.2 Security Levels

The Stryker Badge Cryptographic Module is validated at the FIPS 140-3 section levels shown in the table below.

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	1
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	1
	Overall Level	1

Table 1: Security Levels

2. Cryptographic Module Specification

2.1 Description

2.1.1 Purpose and Use

Stryker Corporation is a global leader in medical technologies, offering innovative products and services in medsurg, neurotechnology, orthopedics, and spine that help improve patient and healthcare outcomes. Stryker's Vocera Badges are wearable communication devices that enable clinician agility and accelerate patient care.

Small and lightweight, the Vocera Badges redefine healthcare communications by bringing together voice calling, secure messaging, alerts, and alarms in a lightweight wearable. Communications are protected via industry-standard secure wireless communications protocols.

The Stryker Badge Cryptographic Module is a firmware module that is installed on the badge, and badge applications can leverage the module's cryptographic services (including encryption/decryption, hashing, digital signature functions, and random number generation) needed to support secure communications.

2.1.2 Module Type

The Stryker Badge Cryptographic Module 3.4.0 is a Firmware module.

2.1.3 Module Embodiment

The Stryker Badge Cryptographic Module has a Multi-Chip Standalone embodiment.

2.1.4 Cryptographic Boundary

The cryptographic boundary is the contiguous perimeter that surrounds all memory-mapped functionality provided by the module when loaded and stored in the host platform's memory.

The cryptographic boundary of the module consists of the following executable file and digest file listed below:

- fips.so (module binary)
- fipsmodule.cnf (digest for module integrity test)

Figure 1 is a block diagram of the module executing in memory and its interactions with surrounding firmware components, as well as the module's cryptographic boundary and Tested Operational Environment's Physical Perimeter (TOEPP).

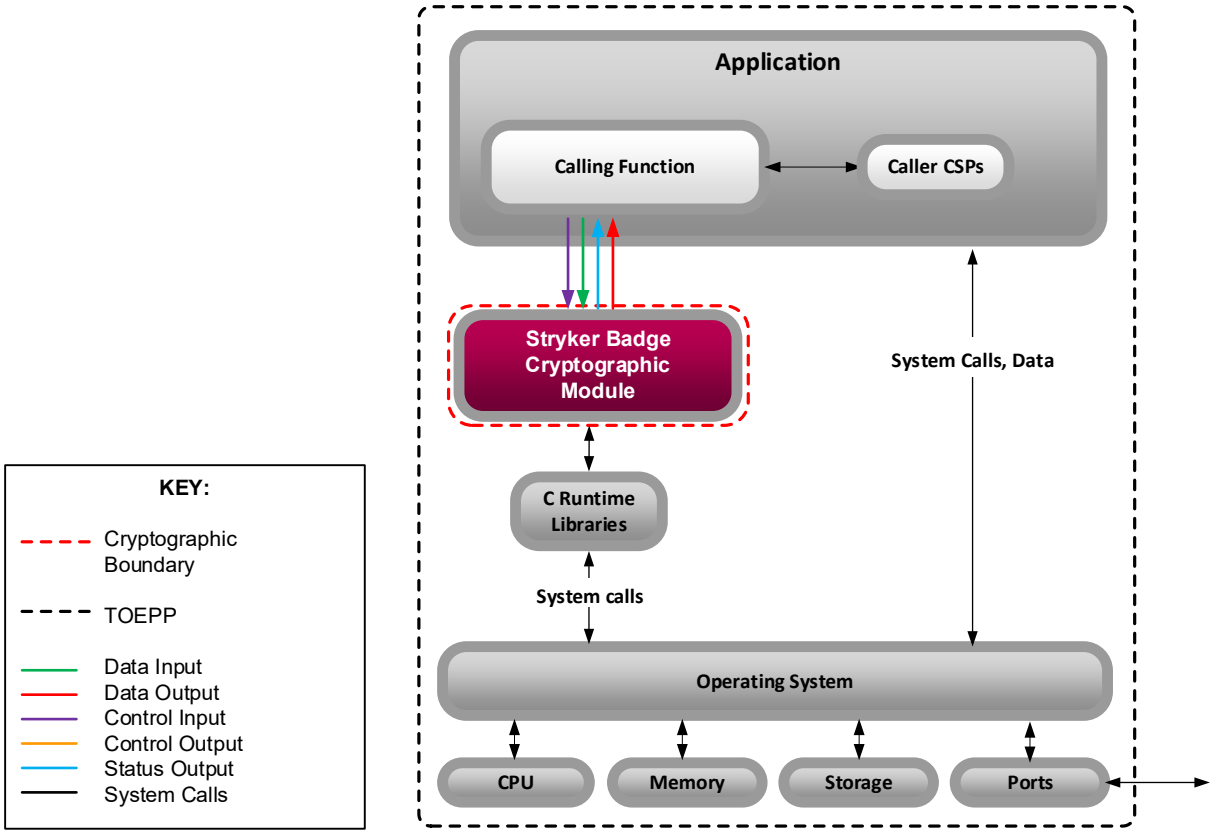

Figure 1: Module Block Diagram

The module is entirely contained within the physical perimeter.

2.1.5 Tested Operational Environment's Physical Perimeter (TOEPP)

As a firmware cryptographic module, the TOEPP of the cryptographic module is defined by each host platform (Stryker's Vocera Badges) on which the module is installed.

2.2 Tested and Vendor Affirmed Module Version and Identification

2.2.1 Tested Module Identification – Hardware

This section is only applicable for hardware modules.

2.2.2 Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

The table below lists the executable code sets of the module.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
fips.so	3.4.0		HMAC-SHA2-256 (Single Encompassing)

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

2.2.3 Tested Module Identification – Hybrid Disjoint Hardware

This section is only applicable to hybrid modules.

2.2.4 Tested Operational Environments – Software, Firmware, Hybrid

The module was tested and found to be compliant with FIPS 140-3 requirements on the environments listed in the table below.

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
BadgeOS 5	V5000	ARM Cortex-A7	No		3.4.0
BadgeOS 7	B7000	ARM Cortex-A53	No		3.4.0

Table 3: Tested Operational Environments - Software, Firmware, Hybrid

2.2.5 Vendor-Affirmed Operational Environments – Software, Firmware, Hybrid

There are no vendor-affirmed operational environments claimed.

2.3 Excluded Components

The module does not exclude any components from the requirements.

2.4 Modes of Operation

2.4.1 Modes List and Description

The table below lists the modes of operation supported by the module.

Mode Name	Description	Type	Status Indicator
Approved	The Approved mode of operation	Approved	EVP_default_properties_is_fips_enabled() returns 1

Table 4: Modes List and Description

2.5 Algorithms

2.5.1 Approved Algorithms

Validation certificates for each Approved algorithm are listed in the table below.

Approved

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A7294	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A7294	Key Length - 128, 192, 256	SP 800-38C
AES-CMAC	A7294	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A7294	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A7294	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A7294	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-KW	A7294	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A7294	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
Counter DRBG	A7294	Prediction Resistance - Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - No, Yes	SP 800-90A Rev. 1
DSA KeyGen (FIPS186-4)	A7294	L - 2048 N - 224, 256	FIPS 186-4
ECDSA KeyGen (FIPS186-5)	A7294	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A7294	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-5)	A7294	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A7294	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-5
Hash DRBG	A7294	Prediction Resistance - Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC DRBG	A7294	Prediction Resistance - Yes Mode - SHA-1, SHA2-256, SHA2-512	SP 800-90A Rev. 1
HMAC-SHA-1	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA2-256	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-256	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A7294	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A7294	Domain Parameter Generation Methods - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KAS-FFC-SSC Sp800-56Ar3	A7294	Domain Parameter Generation Methods - FB, FC, ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192 Scheme - dhEphem - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDF SRTP (CVL)	A7294	AES Key Length - 128, 192, 256	SP 800-135 Rev. 1
KTS-IFC	A7294	Modulo - 2048, 3072, 4096, 6144 Key Generation Methods - rsakpg1-basic , rsakpg1-crt , rsakpg1-prime-factor, rsakpg2-basic , rsakpg2-crt , rsakpg2-prime-factor Scheme - KTS-OAEP-basic - KAS Role - initiator, responder Key Transport Method - Key Length - 1024	SP 800-56B Rev. 2
PBKDF	A7294	Iteration Count - Iteration Count: 1-10000000 Increment 1 Password Length - Password Length: 8-128 Increment 8	SP 800-132
RSA Decryption Primitive Sp800-56Br2 (CVL)	A7294	Modulo - 2048, 3072, 4096	SP 800-56B Rev. 2
RSA KeyGen (FIPS186-5)	A7294	Key Generation Mode - probableWithProbableAux Modulo - 2048, 3072, 4096, 8192 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A7294	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A7294	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
Safe Primes Key Generation	A7294	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	SP 800-56A Rev. 3
Safe Primes Key Verification	A7294	Safe Prime Groups - ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192	SP 800-56A Rev. 3
SHA-1	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA3-224	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-256	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-384	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
SHA3-512	A7294	Message Length - Message Length: 0-65536 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A7294	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
TLS v1.3 KDF (CVL)	A7294	HMAC Algorithm - SHA2-256, SHA2-384 KDF Running Modes - DHE, PSK, PSK-DHE	SP 800-135 Rev. 1

Table 5: Approved Algorithms - Approved

Legacy

Algorithm	CAVP Cert	Properties	Reference
DSA PQGGen (FIPS186-4)	A7294	L - 2048 N - 224, 256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
ECDSA KeyVer (FIPS186-4)	A7294	Curve - B-233, B-283, B-409, B-571, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521	FIPS 186-4
ECDSA SigVer (FIPS186-4)	A7294	Curve - B-163, B-233, B-283, B-409, B-571, K-163, K-233, K-283, K-409, K-571, P-192, P-224, P-256, P-384, P-521 Hash Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-4
RSA SigVer (FIPS186-2)	A7294	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 1536, 2048, 3072, 4096	FIPS 186-4
RSA SigVer (FIPS186-4)	A7294	Signature Type - PKCS 1.5, PKCSPSS Modulo - 1024, 2048, 3072, 4096	FIPS 186-4

Table 6: Approved Algorithms - Legacy

2.5.2 Vendor-Affirmed Algorithms

The vendor affirms the following cryptographic security methods:

- Cryptographic key generation – In compliance with section 4 of *NIST SP 800-133rev2*, the module uses its Approved DRBG to generate random values and seeds used for asymmetric key generation. The generated seed is an unmodified output from the DRBG.

Name	Properties	Implementation	Reference
CKG Section 4	Key Type:Asymmetric	N/A	SP 800-133r2 Section 4, example 1: U is output directly without XORing V

Table 7: Vendor-Affirmed Algorithms

2.5.3 Non-Approved, Allowed Algorithms

The module does not support the use of non-Approved algorithms that are allowed for use in the Approved mode of operation.

N/A for this module.

2.5.4 Non-Approved, Allowed Algorithms with No Security Claimed

The module does not support the use of non-Approved algorithms that are allowed in the Approved mode of operation for which no security is claimed.

N/A for this module.

2.5.5 Non-Approved, Not Allowed Algorithms

The module does not support the use of non-Approved/non-allowed algorithms in the Approved mode of operation.

N/A for this module.

2.6 Security Function Implementations

The table below lists the security function implementations for this module.

Name	Type	Description	Properties	Algorithms
AES for Symmetric Encryption	BC-UnAuthEncrypt	AES Key is used for symmetric encryption		AES-CBC: (A7294) AES-CTR: (A7294) AES-ECB: (A7294)
AES for Symmetric Decryption	BC-UnAuthDecrypt	AES Key is used for symmetric decryption		AES-CBC: (A7294) AES-CTR: (A7294) AES-ECB: (A7294)
AES for Authenticated Symmetric Encryption	BC-AuthEncrypt	AES CCM Key or AES GCM Key is used for authenticated symmetric encryption		AES-GCM: (A7294) AES-CCM: (A7294) AES-KW: (A7294) AES-KWP: (A7294)
AES for Authenticated Symmetric Decryption	BC-AuthDecrypt	AES CCM Key or AES GCM Key is used for authenticated symmetric decryption		AES-GCM: (A7294) AES-CCM: (A7294) AES-KW: (A7294) AES-KWP: (A7294)
AES-CMAC for Message Authentication	MAC	AES CMAC Key is used for generating and verifying symmetric digests		AES-CMAC: (A7294) AES-ECB: (A7294)
DRBG	DRBG	DRBG for generating random bits		Counter DRBG: (A7294) Hash DRBG: (A7294) HMAC DRBG: (A7294)
ECDSA for Key Generation	AsymKeyPair-KeyGen	Used to generate the ECDSA asymmetric key pair		ECDSA KeyGen (FIPS186-5): (A7294) Counter DRBG: (A7294) CKG Section 4: () Key Type: Asymmetric and Symmetric Hash DRBG: (A7294) HMAC DRBG: (A7294)
ECDSA for Key Verification	AsymKeyPair-KeyVer	Used for verifying the ECDSA Public		ECDSA KeyVer (FIPS186-5): (A7294)
ECDSA for Key Verification (Legacy)	AsymKeyPair-KeyVer	Used for verifying the ECDSA public key using legacy parameters	Publication:FIPS 186-4, IG C.M	ECDSA KeyVer (FIPS186-4): (A7294)

Name	Type	Description	Properties	Algorithms
ECDSA for Signature Generation	DigSig-SigGen	ECDSA Private Key is used for generating digital signatures.		ECDSA SigGen (FIPS186-5): (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294) Counter DRBG: (A7294) Hash DRBG: (A7294) HMAC DRBG: (A7294)
ECDSA for Signature Verification	DigSig-SigVer	ECDSA Public Key is used for verifying digital signatures		ECDSA SigVer (FIPS186-5): (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294)
ECDSA for Signature Verification (Legacy)	DigSig-SigVer	ECDSA Public Key is used for verifying digital signatures using legacy parameters		ECDSA SigVer (FIPS186-4): (A7294) SHA-1: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294)
RSA for Key Generation	AsymKeyPair-KeyGen	Used to generate the RSA asymmetric key pair		RSA KeyGen (FIPS186-5): (A7294) Counter DRBG: (A7294) CKG Section 4: () Key Type: Asymmetric and Symmetric Hash DRBG: (A7294) HMAC DRBG: (A7294)
RSA for Signature Generation	DigSig-SigGen	RSA Private Key is used for generating digital signatures		RSA SigGen (FIPS186-5): (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-512: (A7294) Counter DRBG: (A7294)

Name	Type	Description	Properties	Algorithms
RSA for Signature Verification	DigSig-SigVer	RSA Public Key is used for verifying digital signatures		RSA SigVer (FIPS186-5): (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294)
RSA for Signature Verification (Legacy)	DigSig-SigVer	RSA Public Key is used for verifying digital signatures using legacy parameters		RSA SigVer (FIPS186-2): (A7294) RSA SigVer (FIPS186-4): (A7294) SHA-1: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294)
RSA for Key Encapsulation	AsymKeyPair-Encap	RSA-OAEP used for key encapsulation		KTS-IFC: (A7294) RSA KeyGen (FIPS186-5): (A7294) RSA Decryption Primitive Sp800-56Br2: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294) CKG Section 4: () Key Type: Asymmetric Counter DRBG: (A7294) AES-CTR: (A7294) Hash DRBG: (A7294) SHA-1: (A7294) HMAC DRBG: (A7294) HMAC-SHA-1: (A7294) HMAC-SHA2-256: (A7294) HMAC-SHA2-512: (A7294)

Name	Type	Description	Properties	Algorithms
RSA for Key Decapsulation	AsymKeyPair-Decap	RSA-OAEP used for key decapsulation	Publication:SP 800-56B Rev. 2	KTS-IFC: (A7294) RSA KeyGen (FIPS186-5): (A7294) RSA Decryption Primitive Sp800-56Br2: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294) CKG Section 4: () Key Type: Asymmetric Counter DRBG: (A7294) AES-CTR: (A7294) Hash DRBG: (A7294) SHA-1: (A7294) HMAC DRBG: (A7294) HMAC-SHA-1: (A7294) HMAC-SHA2-256: (A7294) HMAC-SHA2-512: (A7294)
SHA for Message Digest	SHA	SHA used for generating message digests		SHA-1: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294)
Safe Primes for Key Generation	AsymKeyPair-KeyGen CKG	Used to generate the DHE asymmetric key pair		Safe Primes Key Generation: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294) Counter DRBG: (A7294) AES-CTR: (A7294) Hash DRBG: (A7294) SHA-1: (A7294) HMAC DRBG: (A7294) HMAC-SHA-1: (A7294) HMAC-SHA2-256: (A7294) HMAC-SHA2-512: (A7294) CKG Section 4: () Key Type: Asymmetric

Name	Type	Description	Properties	Algorithms
Safe Primes for Key Verification	AsymKeyPair-KeyVer	Used to verify the DHE Public Key		Safe Primes Key Verification: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294) Counter DRBG: (A7294) AES-CTR: (A7294) Hash DRBG: (A7294) SHA-1: (A7294) HMAC DRBG: (A7294) HMAC-SHA-1: (A7294) HMAC-SHA2-256: (A7294) HMAC-SHA2-512: (A7294)
HMAC for Message Authentication	MAC	HMAC Key is used for performing keyed hash operations		HMAC-SHA-1: (A7294) HMAC-SHA2-224: (A7294) HMAC-SHA2-256: (A7294) HMAC-SHA2-384: (A7294) HMAC-SHA2-512: (A7294) HMAC-SHA3-224: (A7294) HMAC-SHA3-256: (A7294) HMAC-SHA3-384: (A7294) HMAC-SHA3-512: (A7294) SHA-1: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294)
PBKDF	PBKDF	Used for deriving key from Passphrase		PBKDF: (A7294) SHA-1: (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294)

Name	Type	Description	Properties	Algorithms
DH Shared Secret Computation	CKG KAS-SSC	Used to compute the shared secret "Z"		KAS-FFC-SSC Sp800-56Ar3: (A7294) Safe Primes Key Generation: (A7294) Safe Primes Key Verification: (A7294) DSA KeyGen (FIPS186-4): (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294) CKG Section 4: () Key Type: Asymmetric Counter DRBG: (A7294) DSA PQGGen (FIPS186-4): (A7294)
ECDH Shared Secret Computation	CKG KAS-SSC	Used to compute the shared secret "Z"		KAS-ECC-SSC Sp800-56Ar3: (A7294) ECDSA KeyGen (FIPS186-5): (A7294) ECDSA KeyVer (FIPS186-5): (A7294) SHA2-224: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294) SHA3-224: (A7294) SHA3-256: (A7294) SHA3-384: (A7294) SHA3-512: (A7294) CKG Section 4: () Key Type: Asymmetric Counter DRBG: (A7294)
TLS Key Derivation	KAS-135KDF	TLS v1.2/1.3 KDF used to derive keying material		TLS v1.2 KDF RFC7627: (A7294) TLS v1.3 KDF: (A7294) HMAC-SHA2-256: (A7294) HMAC-SHA2-384: (A7294) HMAC-SHA2-512: (A7294) SHA2-256: (A7294) SHA2-384: (A7294) SHA2-512: (A7294)
SRTP Key Derivation	KAS-135KDF	SRTP KDF used to derive keying material		KDF SRTP: (A7294) AES-ECB: (A7294)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-CTR

The operator is responsible must not reuse a counter value under the same key. Otherwise, the operator may compromise the confidentiality of the ciphertext.

2.7.2 AES-GCM

The module does not implement the TLS protocol. However, the operator may AES GCM in the context of the TLS protocol versions 1.2 and 1.3. To meet the AES GCM (key/IV) pair uniqueness requirements from *NIST SP 800-38D*, the module complies with *FIPS 140-3 IG* C.H as follows:

- For TLS v1.2, the module supports acceptable AES GCM cipher suites from section 3.3.1.1 of *NIST SP 800-52rev2*. The protocol's implementation is contained within the boundary of the module, and the generated IV is only used in the context of the AES GCM encryption executing the provisions of the TLS 1.2 protocol.

  The mechanism for IV generation falls into scenario 1 in *FIPS 140-3 IG* C.H and is compliant with *RFC 5288*. The counter portion of the IV is strictly increasing. When the IV exhausts the maximum number of possible values for a given session key, a failure in encryption will occur and a handshake to establish a new encryption key will be required. It is the responsibility of the module operator (i.e., the first party, client, or server) to trigger this handshake in accordance with *RFC 5246* when this condition is encountered.

- For TLS v1.3, the module supports acceptable AES GCM cipher suites from section 3.3.1.2 of *NIST SP 800-52rev2*. The protocol's implementation is contained within the boundary of the module, and the generated IV is only used in the context of the AES GCM encryption executing the provisions of the TLS 1.3 protocol.

  The mechanism for IV generation falls into scenario 5 in *FIPS 140-3 IG* C.H and is compliant with *RFC 8446*. Each session employs a "per-record nonce", a 64-bit sequence number (or IV) maintained separately for reading and writing records. Each sequence number is set to 0 at the beginning of a connection and whenever the key is changed (the first record transmitted under a particular traffic key uses sequence number 0), and the appropriate sequence number is incremented by one after reading or writing each record. Because the size of sequence numbers is 64 bits, they should not wrap. If a sequence number needs to wrap, it is the responsibility of the module operator to either rekey or terminate the connection.

The module also supports internal IV generation using the module's Approved DRBG. As specified in section 8.2.1 of *NIST SP 800-38D*, internally generated IVs are constructed deterministically, with a length of at least 96 bits. Per *NIST SP 800-38D* and scenario 2 of *FIPS 140-3 IG* C.H, the DRBG generates outputs such that the (key/IV) pair collision probability is less than 2^{-32}.

If power to the module is lost and subsequently restored, the calling application must ensure that any AES GCM keys used for encryption or decryption are re-distributed.

2.7.3 Counter DRBG

Counter DRBG defaults to using a derivation function. However, if the operator chooses to use Counter DRBG without a derivation function, an approved entropy source within the TOEPP is required.

2.7.4 DSA

FIPS 140-3 IG C.K, Resolution 3 allows DSA KeyGen (FIPS186-4) to be used exclusively in a scheme compliant with SP 800-56A Rev. 3. The module prohibits external applications from invoking DSA KeyGen (FIPS186-4) for other purposes.

2.7.5 ECDSA

ECDSA SigVer (FIPS186-4) includes legacy parameters, which include curves with ≤ 112 bits of security strength (B-163, K-163, and P-192) and SHA-1 for the underlying hash algorithm. *FIPS 140-3 IG* C.K, Resolution 6 and Additional Comment 2 allow *FIPS 186-4* parameters to only be used for verifying ECDSA signatures generated prior to February 5, 2024.

2.7.6 RSA

RSA SigVer (FIPS186-2) legacy parameters, which include keys with ≤ 112 bits of security strength (1024 and 1538 bits) and SHA-1 for the underlying hash algorithm. *FIPS 140-3 IG* C.K, Resolution 6 and Additional Comment 2 allow *FIPS 186-2* parameters to only be used for verifying RSA signatures generated prior to September 2020.

RSA SigVer (FIPS186-4) legacy parameters, which include keys with ≤ 112 bits of security strength (1024 bits) and SHA-1 for the underlying hash algorithm. *FIPS 140-3 IG* C.K, Resolution 6 and Additional Comment 2 allow *FIPS 186-4* parameters to only be used for verifying RSA signatures generated prior to February 5, 2024.

2.7.7 KAS

The module performs assurances for its key agreement schemes as specified in the following sections of *NIST SP 800-56Arev3:*

- Section 5.5.2 (for assurances of domain parameter validity)
- Section 5.6.2.1 (for assurances required by the key pair owner)

The module includes the capability to provide the required recipient assurance of ephemeral public key validity specified in section 5.6.2.2.2 of *NIST SP 800-56Arev3*. However, since public keys from other modules are not received directly by this module (those keys are received by the calling application), the module has no knowledge of when a public key is received. Invocation of the proper module services to validate another module's public key is the responsibility of the calling application.

The module implements the following Approved key agreement methods which have been CAVP tested and validated:

- KAS-ECC-SSC (*FIPS 140-3 IG* D.F, Scenario 2 path 1)

- KAS-FFC-SSC (*FIPS 140-3 IG* D.F, Scenario 2 path 1)

The module does not establish SSPs using an approved key agreement scheme (KAS). However, it does offer some or all of the underlying KAS cryptographic functionality to be used by an external operator/application as part of an approved KAS.

2.7.8 KTS

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authentication algorithms that can be used by an external operation/application as part of an approved KTS. Please see Section 2.10.2 for the list of approved authentication algorithms that the module offers.

2.7.9 PBKDF2

The module uses PBKDF2 option 1a from section 5.4 of *NIST SP 800-132*. The iteration count shall be selected as large as possible, as long as the time required to generate the resultant key is acceptable for module operators. The minimum iteration count shall be 1000.

The length of the password/passphrase used in the PBKDF shall be of at least 20 characters, and shall consist of lower-case, upper-case, and numeric characters. The upper bound for the probability of guessing the value is estimated to be $1/62^{20} = 10^{-36}$, which is less than 2^{-112}.

As specified in *NIST SP 800-132*, keys derived from passwords/passphrases may only be used in storage applications.

2.8 RNG and Entropy

The cryptographic module invokes a GET command to obtain entropy for random number generation (the module requests 256 bits of entropy from the calling application per request), and then passively receives entropy from the calling application while having no knowledge of the entropy source and exercising no control over the amount or the quality of the obtained entropy.

The calling application and its entropy sources are located within the operational environment inside the module's physical perimeter but outside the cryptographic boundary. Thus, there is no assurance of the minimum strength of the generated keys.

2.9 Key Generation

The cryptographic module uses its counter-based DRBG to generate seeds used for asymmetric key generation. The generated seed is an unmodified output from the DRBG.

2.10 Key Establishment

2.10.1 Key Agreement Information

The cryptographic module provides the following shared secret computation and key derivation cryptographic primitives necessary to support the DH and ECDH key agreement schemes utilized by the calling application to establish keys.

- KAS-ECC-SSC (Elliptic-curve Diffie-Hellman) with TLS v1.2 KDF RFC7627
- KAS-ECC-SSC (Elliptic-curve Diffie-Hellman) with TLS v1.3 KDF
- KAS-FFC-SSC (Diffie-Hellman) with TLS v1.2 KDF RFC7627
- KAS-FFC-SSC (Diffie-Hellman) with TLS v1.3 KDF

These methods are not used to establish keys into the module.

2.10.2 Key Transport Information

The cryptographic module provides the following cryptographic algorithms necessary to support AES-based key transport mechanisms utilized by the calling application to establish keys.

- Any approved mode of AES with any approved MAC
- AES-CCM
- AES-GCM
- AES-KW
- AES-KWP
- KTS-IFC

These methods are not used to establish keys into the module.

2.11 Industry Protocols

The module uses the following industry protocols:

- SRTP
- TLS 1.2 (with the extended master secret per *RFC 7627*)
- TLS 1.3

No parts of these protocols, other than the Approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.

3. Cryptographic Module Interfaces

3.1 Ports and Interfaces

The module supports the following four logical interfaces:

- Data Input
- Data Output
- Control Input
- Status Output

As a firmware library, the cryptographic module has no direct access to any of the host platform's physical ports, as it communicates only to the calling application via its well-defined API. A mapping of the FIPS-defined interface to the module's physical ports and logical interfaces can be found in the table below.

The module does not output control information and thus has no specified control output interface.

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Logical interface is defined as API input arguments that provide input data for processing. This includes data to be encrypted, decrypted, signed, verified, and hashed, keys to be used in cryptographic services, random seed material for the DRBG of the module, keying material used as input to key establishment services, and intermediate data required for services.
N/A	Data Output	Logical interface is defined as API output arguments that return generated or processed data back to the caller. This includes data that has been encrypted/decrypted/verified, digital signatures, hashes, random values generated by the DRBG of the module, keys established using key establishment methods of the module, and key components/intermediate data/traffic (client and server data and messages).
N/A	Control Input	Logical interface is defined as API input arguments that are used to initialize and control the operation of the module. This includes API commands invoking cryptographic services, modes, key sizes, etc. used with cryptographic services.
N/A	Status Output	Logical interface is defined as API call return values. This includes status information regarding the module or invoked service/operation.
N/A	Power	None.

Table 9: Ports and Interfaces

4. Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication methods; roles are implicitly assumed by the application accessing services implemented by the module.

4.2 Roles

The table below lists the supported roles.

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None
User	Role	User	None

Table 10: Roles

The operator implicitly assumes the Crypto Officer role to execute the “Show status”, “Perform self-tests on demand”, “Zeroize”, and “Show versioning information” services. The User role implicitly assumes the User role to execute the other approved services listed in the table in Section 4.3 below.

The module does not support multiple concurrent operators. The calling application that loaded the module is its only operator.

4.3 Approved Services

This module is a firmware library that provides cryptographic functionality to calling applications. As such, the security functions provided by the module are considered the module’s security services. As allowed per section 2.4.C of *FIPS 140-3 Implementation Guidance*, the module provides indicators for the use of Approved services through a combination of an explicit indication (via a global Approved mode indicator) and an implicit indication (via the successful completion of the service).

The keys and Sensitive Security Parameters (SSPs) listed in the table indicate the type of access required using the following notation:

- G = Generate: The module generates or derives the SSP.
- R = Read: The SSP is read from the module (e.g., the SSP is output).
- W = Write: The SSP is updated, imported, or written to the module.
- E = Execute: The module uses the SSP in performing a cryptographic operation.
- Z = Zeroize: The module zeroizes the SSP.

Descriptions of the services available are provided in the table below.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show status	Shows the status of the module	None	API call	Status output	None	Crypto Officer
Perform self-tests on-demand	Performs the self-tests on demand when invoking the OSSL_PROVIDER_self_test() API command	None	API call	Status output	None	Crypto Officer

Zeroize	Zeroizes the SSPs of the module when calling OPENSSL_cleanse() or re-instantiating the module	None	API call	Status output	None	Crypto Officer - AES Key: Z - AES GCM Key: Z - AES CCM Key: Z - HMAC Key: Z - DHE Private Key: Z - DHE Public Key: Z - ECDHE Private Key: Z - ECDHE Public Key: Z - ECDSA Private Key: Z - ECDSA Public Key: Z - RSA Private Key: Z - RSA Public Key: Z - Passphrase: Z - DRBG Entropy Input: Z - DRBG Seed: Z - DRBG 'Key' Value: Z - DRBG 'C' Value: Z - DRBG 'V' Value: Z - DHE Shared Secret: Z - Derived Keying Material: Z - ECDHE Shared Secret: Z - AES

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
						CMAC Key: Z
Show versioning information	Shows the name and version of the module by executing the OSSL_PROVIDER_get_params() API function	None	API call	Status output	None	Crypto Officer
Perform symmetric encryption	Uses the AES Key to encrypt plaintext into ciphertext	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, AES Key, plaintext	Status output, ciphertext	AES for Symmetric Encryption	User - AES Key: W,E
Perform symmetric decryption	Uses the AES Key to decrypt ciphertext into plaintext	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, AES Key, plaintext	Status output, plaintext	AES for Symmetric Decryption	User - AES Key: W,E
Perform authenticated symmetric encryption	Uses the AES GCM Key or AES CCM Key to encrypt plaintext into ciphertext	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, AES Key, plaintext	Status output, ciphertext	AES for Authenticated Symmetric Encryption	User - AES GCM Key: W,E - AES CCM Key: W,E
Perform authenticated symmetric decryption	Uses the AES GCM Key or AES CCM Key to decrypt ciphertext into plaintext	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, AES Key, plaintext	Status output, plaintext	AES for Authenticated Symmetric Decryption	User - AES GCM Key: W,E - AES CCM Key: W,E
Generate keyed hash (HMAC)	Computes a message authentication code	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, HMAC key, message	Status output, MAC	HMAC for Message Authentication	User - HMAC Key: W,E
Generate symmetric digest (CMAC)	Generates symmetric digest	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, key, plaintext	Status output, digest	AES-CMAC for Message Authentication	User - AES CMAC Key: W,E
Perform key encapsulation	Performs RSA key encapsulation	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, encryption key, key	Status output, encrypted key	RSA for Key Encapsulation	User - RSA Public Key: W,E - Encapsulated Key: R,W
Perform key decapsulation	Performs RSA key decapsulation	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, decryption key, key	Status output, decrypted key	RSA for Key Decapsulation	User - RSA Private Key: W,E - Decapsulated Key: R,W

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Generate asymmetric key pair	Generates a public and private key pair	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters	Status output, key	ECDSA for Key Generation RSA for Key Generation Safe Primes for Key Generation	User - DHE Private Key: G,R - DHE Public Key: G,R - ECDSA Private Key: G,R - ECDSA Public Key: G,R - ECDHE Private Key: G,R - ECDHE Public Key: G,R - RSA Private Key: G,R - RSA Public Key: G,R - DRBG 'Key' Value: E - DRBG 'C' Value: E - DRBG 'V' Value: E
Verify public key	Verifies a DHE, ECDSA, or ECDHE public key	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, key	Status output	ECDSA for Key Verification ECDSA for Key Verification (Legacy) Safe Primes for Key Verification	User - DHE Public Key: W,E - ECDHE Private Key: E - ECDSA Public Key: E
Generate random number	Uses the DRBG to generate random bits	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call	Status output, random bits	DRBG	User - DRBG Entropy Input: W,E - DRBG Seed: G,E - DRBG 'Key' Value: G,E - DRBG 'C' Value: G,E - DRBG 'V' Value: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Generate digital signature	Generates a digital signature using the ECDSA Private Key or RSA Private Key	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, private key, message	Status output, signature	ECDSA for Signature Generation RSA for Signature Generation	User - ECDSA Private Key: W,E - RSA Private Key: W,E - DRBG 'Key' Value: E - DRBG 'C' Value: E - DRBG 'V' Value: E
Verify digital signature	Verifies a digital signature using the ECDSA Public Key or RSA Public Key	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call parameters, public key, signature, message	Status output	ECDSA for Signature Verification ECDSA for Signature Verification (Legacy) RSA for Signature Verification RSA for Signature Verification (Legacy)	User - ECDSA Public Key: W,E - RSA Public Key: W,E
Generate message digest	Uses SHA to hash a message into a message digest	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call, message	Status output, hash	SHA for Message Digest	User
Compute shared secret	Computes DHE/ECDHE shared secret	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call, parameters	Status output, shared secret	DH Shared Secret Computation ECDH Shared Secret Computation	User - DHE Private Key: W,E - DHE Peer Public Key: W,E - ECDHE Private Key: W,E - ECDHE Peer Public Key: W,E - DHE Shared Secret: G,R - ECDHE Shared Secret: G,R

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Derive key via PBKDF2	Uses PBKDF2 to derive key	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call, passphrase	Status output, derived key	PBKDF	User - Passphrase: W,E - Derived Keying Material: G,R
Derive keying material via TLS KDF	Uses TLS v1.2/v1.3 KDF to derive keying material	Global FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call, master secret	Status output, derived key(s).	TLS Key Derivation	User - Derived Keying Material: G,R - Master Secret: W
Derive keying material via SRTP KDF	Uses SRTP KDF to derive keying material	lobal FIPS indicator: EVP_default_properties_is_fips_enabled() returns 1	API call, master secret	Status output, derived key	SRTP Key Derivation	User - Derived Keying Material: G,R - Master Secret: W

Table 11: Approved Services

4.4 Non-Approved Services

The module does not support the use of any non-Approved services.

N/A for this module.

4.5 External Software/Firmware Loaded

The module does not provide the capability to load software from external sources.

5. Software/Firmware Security

5.1 Integrity Techniques

All firmware components within the cryptographic boundary are verified using an Approved integrity technique implemented within the cryptographic module itself. The module implements an HMAC SHA2-256 digest check to test the integrity of the library file; failure of the integrity test for the library file will cause the module to enter a critical error state.

The module's integrity check is performed automatically at module instantiation (i.e., when the module is loaded into memory for execution) without action from the module operator.

5.2 Initiate on Demand

The CO can initiate the pre-operational test on demand by re-instantiating the module or issuing the `OSSL_PROVIDER_self_test()` API command.

6. Operational Environment

6.1 Operational Environment Type and Requirements

The module is a firmware cryptographic library that executes in a Non-Modifiable operational environment.

The cryptographic module has control over its own SSPs. The process and memory management functionality of the host device's OS prevents unauthorized access to plaintext private and secret keys, intermediate key generation values and other SSPs by external processes during module execution. The module only allows access to SSPs through its well-defined API. The operational environment provides the capability to separate individual application processes from each other by preventing uncontrolled access to CSPs and uncontrolled modifications of SSPs regardless of whether this data is in the process memory or stored on persistent storage within the operational environment. Processes that are spawned by the module are owned by the module and are not owned by external processes/operators.

7. Physical Security

The Stryker Badge Cryptographic Module is a multi-chip standalone firmware module contained within a production-grade enclosure that executes on Stryker's Vocera Badge hardware devices. All components of these devices are made of production-grade materials, and all integrated circuits are coated with commercial standard passivation.

8. Non-Invasive Security

This section is not applicable. There are currently no approved non-invasive mitigation techniques references in Annex F of *ISO/IEC 19790*.

9. Sensitive Security Parameters Management

9.1 Storage Areas

There are no mechanisms within the module's cryptographic boundary for the persistent storage of SSPs. The module stores DRBG state values for the lifetime of the DRBG instance. The module uses SSPs passed in on the stack by the calling application and does not store these SSPs beyond the lifetime of the API call.

The table below lists sensitive security parameters (SSPs) storage areas for this module.

Storage Area Name	Description	Persistence Type
RAM	SSPs are stored in RAM temporarily	Dynamic

Table 12: Storage Areas

9.2 SSP Input-Output Methods

The table below lists input and output methods for the module's SSPs. Section 9.4 below selects from the input and output methods listed and specifies the appropriate parameter in the "Inputs/Outputs" column if applicable to a specific SSP.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
API call input	External	RAM	Plaintext	Manual	Electronic	
API call output	RAM	External	Plaintext	Manual	Electronic	

Table 13: SSP Input-Output Methods

9.3 SSP Zeroization Methods

The table below lists SSP zeroization methods for this module Section 9.4 below selects from the zeroization methods listed and specifies the appropriate parameter in the "Zeroization" column if applicable to a specific SSP.

Zeroization Method	Description	Rationale	Operator Initiation
OPENSSL_cleanse()	OPENSSL_cleanse() API call zeroizes SSPs	OPENSSL_cleanse() API call zeroizes SSPs by replacing the pointer of the specified length with 0's, yielding the SSP irretrievable	The operator calls OPENSSL_cleanse()
Reboot	Rebooting the host device zeroizes SSPs	Rebooting the device clears SSPs in RAM, yielding them irretrievable	The operator reboots the host device

Table 14: SSP Zeroization Methods

9.4 SSPs

The module supports the keys and other SSPs listed in the table below. All SSP imports and exports are electronic and performed within the TOEPP.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	Used for symmetric encryption/decryption	128 - 256 - 128 - 256	Symmetric Key - CSP			AES for Symmetric Encryption AES for Symmetric Decryption
AES GCM Key	Used for authenticated symmetric encryption/decryption	128 - 256 - 128 - 256	Symmetric Key - CSP			AES for Authenticated Symmetric Encryption AES for Authenticated Symmetric Decryption
AES CCM Key	Used for authenticated symmetric encryption/decryption	128 - 256 - 128 - 256	Symmetric Key - CSP			AES for Authenticated Symmetric Encryption AES for Authenticated Symmetric Decryption
AES CMAC Key	Used for generating and verifying message authentication	128 - 256 - 128 - 256	Authentication - CSP			AES-CMAC for Message Authentication
HMAC Key	Used for keyed hash messages	160 - 512 - 160 - 512	Authentication - CSP			HMAC for Message Authentication
DHE Private Key	Used for DHE shared secret computation	[ffdhe] Between 224 and 400 bits [FB/FC] Between 224 and 256 bits [ffdhe] Between 2048 and 8192 bits - [ffdhe] Between 112 and 200 bits [FB/FC] Between 112 and 128 bits [ffdhe] Between 112 and 200 bits	Private - CSP	Safe Primes for Key Generation DSA KeyGen (FIPS186-4) (A7294)		DH Shared Secret Computation
DHE Public Key	Sent to peer for DHE shared secret computation	[ffdhe] Between 2048 and 8192 bits [FB/FC] 2048 bits - [ffdhe] Between 112 and 200 bits [FB/FC] 112 bits	Public - PSP	Safe Primes for Key Generation DSA KeyGen (FIPS186-4) (A7294)		DH Shared Secret Computation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DHE Peer Public Key	Used for DHE shared secret computation.	[ffdhe] Between 2048 and 8192 bits [FB/FC] 2048 bits - [ffdhe] Between 112 and 200 bits [FB/FC] 112 bits	Public - PSP			
DHE Shared Secret	Shared secret derived from the DHE Private Key and the DHE Public Key.	[ffdhe] Between 2048 and 8192 bits [FB/FC] 2048 bits - [ffdhe] Between 112 and 200 bits [FB/FC] 112 bits	Shared Secret - CSP		DH Shared Secret Computation	
ECDHE Private Key	Used for ECDHE shared secret computation	Between 224 and 571 bits - Between 112 and 256 bits	Private - CSP	ECDSA for Key Generation		ECDH Shared Secret Computation
ECDHE Public Key	Sent to peer for DHE shared secret computation	Between 224 and 571 bits - Between 112 and 256 bits	Public - PSP	ECDSA for Key Generation		
ECDHE Peer Public Key	Used for ECDHE shared secret computation.	Between 224 and 571 bits - Between 112 and 256 bits	Public - PSP			ECDH Shared Secret Computation
ECDHE Shared Secret	Shared secret derived from the ECDHE Private Key and the ECDHE Public Key.	Between 224 and 571 bits - Between 112 and 256 bits	Shared Secret - CSP		ECDH Shared Secret Computation	
ECDSA Private Key	Used for ECDSA digital signature generation	Between 224 and 571 bits - Between 112 and 256 bits	Private - CSP	ECDSA for Key Generation		ECDSA for Signature Generation
ECDSA Public Key	Used for ECDSA digital signature verification	Between 163 and 571 bits - Between 80 and 256 bits	Public - PSP	ECDSA for Key Generation		ECDSA for Signature Verification ECDSA for Signature Verification (Legacy)
RSA Private Key	Used for RSA digital signature generation and key decapsulation	[RSA SigGen] Between 2048 and 4096 bits [KTS-IFC] Between 2048 and 6144 bits - RSA SigGen] Between 112 and 150 bits [KTS-IFC] Between 112 and 176 bits	Private - CSP	RSA for Key Generation		RSA for Signature Generation RSA for Key Decapsulation
RSA Public Key	Used for RSA digital signature verification and key encapsulation	[RSA SigVer] Between 1024 and 4096 bits [KTS-IFC] Between 2048 and 6144 bits - [RSA SigVer] Between 80 and 150 bits [KTS-IFC] Between 112 and 176 bits	Public - PSP	RSA for Key Generation		RSA for Signature Verification RSA for Key Encapsulation

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
Master Secret	Used to generate Derived Keying Material using SRTF KDF or TLS KDF	384 bits - 384 bits	Shared Secret - CSP			TLS Key Derivation SRTP Key Derivation
Derived Keying Material	Keying material derived from a key derivation function	Between 128 and 256 bits - Between 128 and 256 bits	Shared Secret - CSP	PBKDF TLS Key Derivation SRTP Key Derivation		
Encapsulated Key	Encapsulated by RSA Public Key.	Between 112 and 5680 bits - Between 112 and 256 bits	Key - CSP			
Decapsulated Key	Decapsulated by the RSA Private Key.	Between 112 and 5680 bits - Between 112 and 256 bits	Key - CSP			
Passphrase	Used as input for PBKDF	Between 64 and 1024 bits - Between 64 and 1024 bits	Passphrase - CSP			PBKDF
DRBG Entropy Input	Entropy input used for the DRBG Seed	[Counter DRBG] Between 128 and 512 bits [Hash DRBG] Between 128 and 320 bits [HMAC DRBG] Between 160 and 1024 bits - [Counter DRBG] Between 128 and 512 bits [Hash DRBG] Between 128 and 320 bits [HMAC DRBG] Between 160 and 1024 bits	Entropy Input - CSP			DRBG
DRBG Seed	Seed used for the DRBG	DRBG] Between 256 and 384 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 440 or 888 bits - DRBG] Between 256 and 384 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 440 or 888 bits	Seed - CSP	DRBG		DRBG
DRBG 'Key' Value	DRBG state value	[Counter DRBG] Between 128 and 256 bits [HMAC DRBG] 160, 256, or 512 bits - [Counter DRBG] Between 128 and 256 bits [HMAC DRBG] 160, 256, or 512 bits	State Value - CSP	DRBG		DRBG
DRBG 'C' Value	DRBG state value	440 or 888 bits - 440 or 888 bits	State Value - CSP	DRBG		DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG 'V' Value	DRBG state value	[Counter DRBG] 128 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 160, 256, or 512 bits - [Counter DRBG] 128 bits [Hash DRBG] 440 or 888 bits [HMAC DRBG] 160, 256, or 512 bits	State Value - CSP	DRBG		DRBG

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
AES GCM Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
AES CCM Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
AES CMAC Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
HMAC Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
DHE Private Key	API call input API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DHE Public Key:Paired With DHE Shared Secret:Derives DHE Peer Public Key:Used With
DHE Public Key	API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DHE Private Key:Paired With
DHE Peer Public Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DHE Private Key:Used With DHE Shared Secret:Derives
DHE Shared Secret	API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DHE Private Key:Derived From DHE Peer Public Key:Derived From
ECDHE Private Key	API call input API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	ECDHE Public Key:Paired With ECDHE Public Key:Used With ECDHE Shared Secret:Derives

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
ECDHE Public Key	API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	ECDHE Private Key:Paired With
ECDHE Peer Public Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	ECDHE Private Key:Used With ECDHE Shared Secret:Derives
ECDHE Shared Secret	API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted.	OPENSSL_cleanse() Reboot	ECDHE Private Key:Derived From ECDHE Peer Public Key:Derived From
ECDSA Private Key	API call input API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	ECDSA Public Key:Paired With
ECDSA Public Key	API call input API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	ECDSA Private Key:Paired With
RSA Private Key	API call input API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	RSA Public Key:Paired With Decapsulated Key:Decrypts
RSA Public Key	API call input API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	RSA Private Key:Paired With Encapsulated Key:Encrypts
Master Secret	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
Derived Keying Material	API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	Master Secret:Derived From Passphrase:Derived From
Encapsulated Key	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	RSA Public Key:Encrypted By
Decapsulated Key	API call output	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	RSA Private Key:Decrypted By
Passphrase	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
DRBG Entropy Input	API call input	RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	
DRBG Seed		RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DRBG Entropy Input:Derived From
DRBG 'Key' Value		RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DRBG Seed:Derived From
DRBG 'C' Value		RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DRBG Seed:Derived From

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
DRBG 'V' Value		RAM:Plaintext	Stored in RAM until OPENSSL_cleanse() is called or the host device is rebooted	OPENSSL_cleanse() Reboot	DRBG Seed:Derived From

Table 16: SSP Table 2

9.5 Transitions

The following list specifies applicable transition periods or timeframes where an algorithm or key length transitions from Approved to non-Approved:

- Key Sizes: In compliance with *NIST SP 800-131A Rev. 2*, the module supports algorithms and key lengths that provide a minimum of 112 bits of security strength for applying cryptographic protection. Starting January 1, 2031, the minimum security strength for applying cryptographic protection will be 128 bits, and security strengths between 112 bits and 128 bits will be allowed for legacy use only to process information that is already protected.

- SHA-1: The module offers support for SHA-1 for hashing. This implementation will be non-Approved for all uses starting January 1, 2031.

10. Self-Tests

10.1 Pre-Operational Self-Tests

The module performs the pre-operational self-tests listed in the following table.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256	SHA2-256	SW/FW Integrity	SW/FW Integrity	Returns 1 if the test succeeds. Returns 0 if the test fails	HMAC-SHA2-256 integrity test on fips.so

Table 17: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

The module performs the conditional self-tests listed in the following table.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-GCM Decrypt	256-bit	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Decrypt	After pre-operational integrity test
AES-GCM Encrypt	256-bit	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Encrypt	After pre-operational integrity test
AES-ECB	128-bit	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Decrypt	After pre-operational integrity test
ECDSA SigGen (FIPS186-5) #1	B-233; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Sign	After pre-operational integrity test
ECDSA SigVer (FIPS186-5) #1	B-233; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Verify	After pre-operational integrity test
ECDSA SigGen (FIPS186-5) #2	P-224; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Sign	After pre-operational integrity test
ECDSA SigVer (FIPS186-5) #2	P-224; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Verify	After pre-operational integrity test
ECDSA SigVer (Legacy)	P-192; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Verify	After pre-operational integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
RSA SigGen (FIPS186-5)	2048-bits; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Sign	After pre-operational integrity test
RSA SigVer (FIPS186-5)	2048-bits; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Verify	After pre-operational integrity test
RSA SigVer (Legacy)	1024/1536-bits; SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Verify	After pre-operational integrity test
Counter DRBG	AES-128-CTR; with derivation function;	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Generate/Initiate/Reseed	After pre-operational integrity test
Hash DRBG	SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Generate/Initiate/Reseed	After pre-operational integrity test
HMAC DRBG	SHA-1	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Generate/Initiate/Reseed	After pre-operational integrity test
SHA2-512	SHA2-512	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Digest	After pre-operational integrity test
SHA3-256	SHA3-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Digest	After pre-operational integrity test
HMAC-SHA2-256	SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Hashed Message	After pre-operational integrity test
KAS-ECC-SSC Sp800-56Ar3	P-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Shared Secret Computation	After pre-operational integrity test
KAS-FFC-SSC Sp800-56Ar3	ffdhe2048	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Shared Secret Computation	After pre-operational integrity test
PBKDF	SHA2-256; 4096 iterations	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Derive	After pre-operational integrity test
KTS-IFC	2048-bits	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Encrypt	After pre-operational integrity test
TLS v1.2 KDF RFC7627	SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Derive	After pre-operational integrity test

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
TLS v1.3 KDF	SHA2-256	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Derive	After pre-operational integrity test
KDF SRTP	AES-128-CTR	KAT	CAST	Returns 1 if the test succeeds. Returns 0 if the test fails	Derive	After pre-operational integrity test
ECDSA KeyGen (FIPS186-5)		PCT	PCT	Returns 1 if the test succeeds. Returns 0 if the test fails	Sign/Verify	When the requested service requires the generation of an ECDH/ECDSA key pair
RSA KeyGen (FIPS186-5)		PCT	PCT	Returns 1 if the test succeeds. Returns 0 if the test fails	Sign/Verify	When the requested service requires the generation of an RSA key pair
DSA KeyGen (FIPS186-4)		PCT	PCT	Returns 1 if the test succeeds. Returns 0 if the test fails	Key Generation	When the requested service requires the generation of a DHE key pair
Safe Primes Key Generation		PCT	PCT	Returns 1 if the test succeeds. Returns 0 if the test fails	Key Generation	When the requested service requires the generation of an DHE key pair

Table 18: Conditional Self-Tests

10.3 Periodic Self-Test Information

The table below specifies the module's periodic self-test information.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256	SW/FW Integrity	SW/FW Integrity	On Demand	Manually

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-GCM Decrypt	KAT	CAST	On Demand	Manually
AES-GCM Encrypt	KAT	CAST	On Demand	Manually
AES-ECB	KAT	CAST	On Demand	Manually
ECDSA SigGen (FIPS186-5) #1	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-5) #1	KAT	CAST	On Demand	Manually
ECDSA SigGen (FIPS186-5) #2	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-5) #2	KAT	CAST	On Demand	Manually
ECDSA SigVer (Legacy)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-5)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-5)	KAT	CAST	On Demand	Manually
RSA SigVer (Legacy)	KAT	CAST	On Demand	Manually
Counter DRBG	KAT	CAST	On Demand	Manually
Hash DRBG	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC DRBG	KAT	CAST	On Demand	Manually
SHA2-512	KAT	CAST	On Demand	Manually
SHA3-256	KAT	CAST	On Demand	Manually
HMAC-SHA2-256	KAT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3	KAT	CAST	On Demand	Manually
KAS-FFC-SSC Sp800-56Ar3	KAT	CAST	On Demand	Manually
PBKDF	KAT	CAST	On Demand	Manually
KTS-IFC	KAT	CAST	On Demand	Manually
TLS v1.2 KDF RFC7627	KAT	CAST	On Demand	Manually
TLS v1.3 KDF	KAT	CAST	On Demand	Manually
KDF SRTP	KAT	CAST	On Demand	Manually
ECDSA KeyGen (FIPS186-5)	PCT	PCT		
RSA KeyGen (FIPS186-5)	PCT	PCT		
DSA KeyGen (FIPS186-4)	PCT	PCT		
Safe Primes Key Generation	PCT	PCT		

Table 20: Conditional Periodic Information

10.4 Error States

The table below specifies the module's error state information.

Name	Description	Conditions	Recovery Method	Indicator
Critical Error	When the module enters this error state, the module inhibits all data output	When the module fails the power-up integrity test, KATs, or conditional PCTs	To recover from the Critical Error state, please power cycle the module. If the error persists after power cycling, the operator should contact Stryker Corporation for assistance	EVP_get_error() returns PROV_R_FIPS_MODULE_IN_ERROR_STATE

Table 21: Error States

If the module continues to experience self-test failures after reinitializing, then the module will not be able to resume normal operations, and the CO should contact Stryker Corporation for assistance.

10.5 Operator Initiation of Self-Tests

The CO can initiate the pre-operational self-tests and conditional CASTs on demand by re-instantiating the module or issuing the `OSSL_PROVIDER_self_test()` API command.

11. Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

The Stryker Badge Cryptographic Module is not delivered to end-users as a standalone offering. Rather, it is a pre-built, integrated component of the Stryker badge application firmware, and these applications are the sole consumers of the cryptographic services provided by the module.

The badge application firmware is delivered pre-installed on Stryker Badges. Stryker does not provide end-users with any mechanisms to directly access the module, its source code, its APIs, or any information sent to/from the module.

There are no end-user procedures for module startup, and the application firmware starts when power is applied to the badge. The module's integrity check is performed automatically at module startup (i.e., when the module is loaded into memory for execution) without action from the module operator, and end-users have no ability to bypass the automatic integrity check.

No setup steps are required to be performed by end-users.

11.2 Administrator Guidance

The `OSSL_PROVIDER_get_params()` API provides access to the current status of the module, as well as the name and version.

- The `OSSL_PROVIDER_get_params()` API can be used to determine the module's operational status. A return value of '1' indicates that the module has passed all pre-operational self-tests and is currently in its Approved mode; a return value of '0' indicates an error.

- The `OSSL_PROVIDER_get_params()` API can be used to obtain the module's versioning information. The API takes in OSSL_PARAM structure as an argument. This structure contains the "key" variable, which corresponds with the name of the parameter. If the OSSL_PARAM structure's "key" variable is "name", the API will return the module name. If the OSSL_PARAM structure's "key" variable is "version", the API will return the module version. Both values can be correlated with the module's validation record.

  The expected output of `OSSL_PROVIDER_get_params()` when the "key" variable is "name" is **Stryker Badge Cryptographic Module**; the expected output of `OSSL_PROVIDER_get_params()` when the "key" variable is "version" is **3.4.0**.

The module is affected by CVE-2026-31790. To mitigate this vulnerability, the calling application must invoke `EVP_PKEY_public_check()` or `EVP_PKEY_public_check_quick()` to validate the public key before calling `EVP_PKEY_encapsulate()`.

If any irregular activity is observed, or if the module is consistently reporting errors, then Stryker Customer Support should be contacted.

11.3 Non-Administrator Guidance

There is no specific guidance for module operators acting in a non-administrator capacity.

11.4 Design and Rules

The module is a cryptographic library used by a calling application. The calling application is responsible for:

- Use of the primitives in the correct sequence.
- Use of keys in accordance with *NIST SP 800-140Drev2* (as the keys used by the module for cryptographic purposes are provided over the call stack by the calling application).
- Use of a *NIST SP 800-90B* compliant entropy source outside the module boundary with at least 256 bits of security strength. Entropy is supplied to the module via callback functions. The callback functions shall return an error if the minimum entropy strength cannot be met.

12. Mitigation of Other Attacks

12.1 Attack List

The Module implements mitigations for constant-time implementations and blinding attacks.

12.2 Mitigation Effectiveness

Constant-time implementations protect cryptographic implementations in the Module against timing analysis since such attacks exploit differences in execution time depending on the cryptographic operation, and constant-time implementations ensure that the variations in execution time cannot be traced back to the key, CSP, or secret data.

Numeric blinding protects the RSA, DSA and ECDSA algorithms from timing attacks. These algorithms are vulnerable to such attacks since attackers can measure the time of signature operations or RSA decryption. To mitigate this, the Module generates a random blinding factor which is provided as an input to the decryption/signature operation and is discarded once the operation has completed and resulted in an output. This makes it difficult for attackers to attempt timing attacks on such operations without the knowledge of the blinding factor, and therefore the execution time cannot be correlated to the RSA/DSA/ECDSA key.

12.3 Guidance and Constraints

The mitigation mechanisms described in Section 12.2 are inherent within the validated algorithms. No other guidance or constraints are specified.

Appendix A. Acronyms and Abbreviations

Table 22 provides definitions for the acronyms and abbreviations used in this document.

Table 22: Acronyms and Abbreviations

Term	Definition
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CBC	Cipher Block Chaining
CCCS	Canadian Centre for Cyber Security
CCM	Counter with Cipher Block Chaining - Message Authentication Code
CFB	Cipher Feedback
CKG	Cryptographic Key Generation
CMAC	Cipher-Based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CO	Cryptographic Officer
CPU	Central Processing Unit
CSP	Critical Security Parameter
CTR	Counter
CVL	Component Validation List
DEP	Default Entry Point
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
HMAC	(keyed-) Hash Message Authentication Code
KAS	Key Agreement Scheme

Term	Definition
KAT	Known Answer Test
KDF	Key Derivation Function
KTS	Key Transport Scheme
KW	Key Wrap
KWP	Key Wrap with Padding
NIST	National Institute of Standards and Technology
OCB	Offset Codebook
OFB	Output Feedback
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standard
PSS	Probabilistic Signature Scheme
PUB	Publication
RFC	Request for Comment
RNG	Random Number Generator
RSA	Rivest Shamir Adleman
SHA	Secure Hash Algorithm
SHAKE	Secure Hash Algorithm KECCAK
SHS	Secure Hash Standard
SRTP	Secure Real-time Transport Protocol
SP	Special Publication
TLS	Transport Layer Security

Prepared by:
Corsec Security, Inc.

12600 Fair Lakes Circle, Suite 210
Fairfax, VA 22033
United States of America

Phone: +1 703 267 6050
Email: info@corsec.com
http://www.corsec.com