Zimperium, Inc.

Zimperium zKeyBox white-box cryptography library

FIPS 140-3 Non-Proprietary Security Policy

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Table of Contents

1 General 5
1.1 Overview 5
1.2 Security Levels 5
1.3 Additional Information 5
2 Cryptographic Module Specification 5
2.1 Description 5
2.2 Tested and Vendor Affirmed Module Version and Identification 7
2.3 Excluded Components 8
2.4 Modes of Operation 8
2.5 Algorithms 9
2.6 Security Function Implementations 12
2.7 Algorithm Specific Information 13
2.8 RBG and Entropy 14
2.9 Key Generation 15
2.10 Key Establishment 16
2.11 Industry Protocols 16
3 Cryptographic Module Interfaces 16
3.1 Ports and Interfaces 16
4 Roles, Services, and Authentication 16
4.1 Authentication Methods 16
4.2 Roles 17
4.3 Approved Services 17
4.4 Non-Approved Services 23
4.5 External Software/Firmware Loaded 23
5 Software/Firmware Security 23
5.1 Integrity Techniques 23
5.2 Initiate on Demand 24
6 Operational Environment 24
6.1 Operational Environment Type and Requirements 24
7 Physical Security 24
8 Non-Invasive Security 24
9 Sensitive Security Parameters Management 24
9.1 Storage Areas 24
9.2 SSP Input-Output Methods 24
9.3 SSP Zeroization Methods 25

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

9.4 SSPs 25
10 Self-Tests 27
10.1 Pre-Operational Self-Tests 27
10.2 Conditional Self-Tests 27
10.3 Periodic Self-Test Information 30
10.4 Error States 31
10.5 Operator Initiation of Self-Tests 32
11 Life-Cycle Assurance 32
11.1 Installation, Initialization, and Startup Procedures 32
11.2 Administrator Guidance 32
11.3 Non-Administrator Guidance 32
11.4 Design and Rules 32
11.5 Additional Information 33
12 Mitigation of Other Attacks 33

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

List of Tables

Table 1: Security Levels 5
Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets) 8
Table 3: Tested Operational Environments - Software, Firmware, Hybrid 8
Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid 8
Table 5: Modes List and Description 8
Table 6: Approved Algorithms 11
Table 7: Vendor-Affirmed Algorithms 11
Table 8: Security Function Implementations 13
Table 9: Ports and Interfaces 16
Table 10: Roles 17
Table 11: Approved Services 23
Table 12: Storage Areas 24
Table 13: SSP Input-Output Methods 25
Table 14: SSP Zeroization Methods 25
Table 15: SSP Table 1 27
Table 16: SSP Table 2 27
Table 17: Pre-Operational Self-Tests 27
Table 18: Conditional Self-Tests 29
Table 19: Pre-Operational Periodic Information 30
Table 20: Conditional Periodic Information 31
Table 21: Error States 31

List of Figures

Figure 1: Block Diagram 7

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

1 General

1.1 Overview

This document comprises the non-proprietary FIPS 140-3 Security Policy for the Zimperium zKeyBox white-box cryptography library version 5.38.1. In this document the Zimperium zKeyBox white-box cryptography library is referred to as SKB, the library, or the module. The overall security level of the module is 1.

1.2 Security Levels

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	N/A
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	N/A
	Overall Level	1

Table 1: Security Levels

1.3 Additional Information

FIPS 140-3 (Federal Information Processing Standards Publication 140-3 – Security Requirements for Cryptographic Modules) 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 National Institute of Standards and Technology (NIST) and Canadian Centre for Cyber Security (CCCS): https://csrc.nist.gov/projects/cryptographic-module-validation-program.

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Zimperium zKeyBox white-box cryptography library, also referred to as the SKB or zKeyBox, is a C/C++ library that provides an extensive set of high-level classes and methods for working with and containing the most popular cryptographic algorithms. The library's unique white-box implementation is specifically designed to hide and protect cryptographic keys at all

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

times. In SKB, keys are always obfuscated, and cryptographic algorithms operate directly with obfuscated keys.

SKB has a comprehensive interface that provides access to a set of cryptographic algorithms. An application that is integrated with SKB executes the cryptographic algorithms via the zKeyBox API.

Because of the library's robust design, systems integrated with SKB can be safely deployed in insecure environments, such as mobile devices, tablets, and desktop computers, where anyone could have access to the program code and device memory.

Module Type: Software

Module Embodiment: Multi-Chip Standalone

Cryptographic Boundary:

The cryptographic boundary of the module is the Zimperium zKeyBox white-box cryptography library. The module is comprised of a single shared object (SO). The module performs no communications other than with the calling application (the process that invokes the module services).

Tested Operational Environment's Physical Perimeter (TOEPP):

The module's TOEPP is the physical perimeter of the tested platforms listed in Table "Tested Operational Environments – Software, Firmware, Hybrid" below.

The module's block diagram is shown in Figure 1 below. The red border in the figure denotes the cryptographic boundary of the module. The outer black border denotes the TOEPP of the module.

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

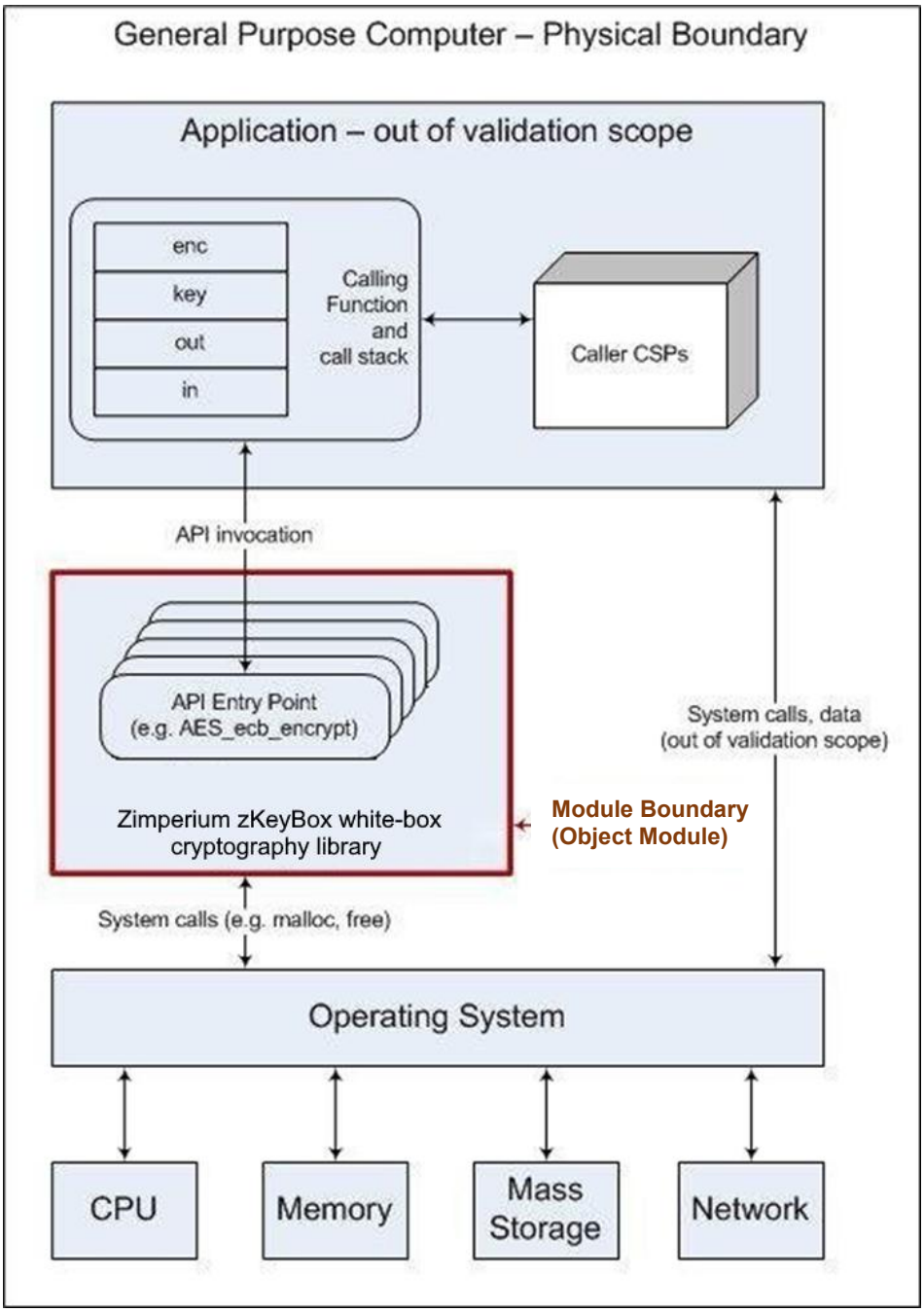


Figure 1: Block Diagram

2.2 Tested and Vendor Affirmed Module Version and Identification

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

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
libSecureKeyBoxFips.so	5.38.1	Library file for Android platform	HMAC-SHA2-256
SecureKeyBoxFips	5.38.1	Library file for iOS platform	HMAC-SHA2-256

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

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Android 13	Google Pixel 5	Qualcomm Snapdragon 765G	No		5.38.1
iOS 16	iPhone 12 mini	Apple A14	No		5.38.1

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

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform
Windows 10, 11	General PC
MacOS 12, 13, 14, 15	General PC
Ubuntu 18.04, 20.04, 22.04, 24.04	General PC

Table 4: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

There are no components excluded from the module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Types of Services	Status Indicator
Approved Mode	The Module operates only in approved mode	Approved	SKB_FIPS_GetErrorStatus() returns global status indicator '0' when the module is operating in Approved mode

Table 5: Modes List and Description

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

The module only supports approved mode of operation. Please see Section 11.1 for initialization procedure.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A4111	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A4111	Key Length - 128, 192, 256 Tag Length - 112, 128, 32, 48, 64, 80, 96 IV Length - IV Length: 56, 64, 72, 80, 88, 96, 104 Payload Length - Payload Length: 0-256 Increment 8 AAD Length - AAD Length: 0-524288 Increment 8	SP 800-38C
AES-CMAC	A4111	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 0-65536 Increment 8	SP 800-38B
AES-CMAC	A4112	Direction - Generation, Verification Key Length - 128, 192, 256 MAC Length - MAC Length: 128 Message Length - Message Length: 0-65536 Increment 8	SP 800-38B
AES-CTR	A4111	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - Yes Incremental Counter - Yes Counter Tests Performed - Yes	SP 800-38A
AES-ECB	A4111	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4112	Direction - Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A4113	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A4111	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.2 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96	SP 800-38D

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Algorithm	CAVP Cert	Properties	Reference
		IV Length - IV Length: 96 Payload Length - Payload Length: 0-65536 Increment 8 AAD Length - AAD Length: 0-65536 Increment 256	
AES-GMAC	A4111	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.2 Key Length - 128, 192, 256 Tag Length - 104, 112, 120, 128, 32, 64, 96 IV Length - IV Length: 96 AAD Length - AAD Length: 256-65536 Increment 256	SP 800-38D
AES-KW	A4111	Direction - Decrypt, Encrypt Cipher - Cipher Key Length - 128, 192, 256 Payload Length - Payload Length: 128-4096 Increment 64	SP 800-38F
ECDSA SigGen (FIPS186-4)	A4111	Component - No Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-4
Hash DRBG	A4111	Prediction Resistance - No Supports Reseed - Yes Mode - SHA2-256 Entropy Input - Entropy Input: 256-512 Increment 8 Nonce - Nonce: 128-256 Increment 8 Personalization String Length - Personalization String Length: 0-256 Increment 8 Additional Input - Additional Input: 0 Returned Bits - 256	SP 800-90A Rev. 1
HMAC-SHA2-224	A4111	MAC - MAC: 32-224 Increment 8 Key Length - Key Length: 8-4096 Increment 8	FIPS 198-1
HMAC-SHA2-256	A4111	MAC - MAC: 32-256 Increment 8 Key Length - Key Length: 8-4096 Increment 8	FIPS 198-1
HMAC-SHA2-384	A4111	MAC - MAC: 32-384 Increment 8 Key Length - Key Length: 8-4096 Increment 8	FIPS 198-1
HMAC-SHA2-512	A4111	MAC - MAC: 32-512 Increment 8 Key Length - Key Length: 8-4096 Increment 8	FIPS 198-1

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Algorithm	CAVP Cert	Properties	Reference
KAS-ECC CDH-Component SP800-56Ar3 (CVL)	A4111	Curve - P-224, P-256, P-384, P-521	SP 800-56A Rev. 3
KDF SP800-108	A4111	KDF Mode - Counter MAC Mode - CMAC-AES128 Supported Lengths - Supported Lengths: 128-4096 Increment 128 Fixed Data Order - Before Fixed Data Counter Length - 8 Supports Empty IV - No Requires Empty IV - No Custom Key In Length - 0	SP 800-108 Rev. 1
RSA SigGen (FIPS186-4)	A4111	Signature Type - PKCS 1.5, PKCSPSS Modulo - 2048, 3072, 4096 Hash Pair - Hash Algorithm - SHA2-224	FIPS 186-4
SHA2-224	A4111	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-224	A4112	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4111	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-256	A4112	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4111	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-384	A4112	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4111	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4
SHA2-512	A4112	Message Length - Message Length: 0-65536 Increment 8	FIPS 180-4

Table 6: Approved Algorithms

Vendor-Affirmed Algorithms:

Name	Properties	Implementation	Reference
CKG	Key Type:Symmetric	N/A	NIST SP 800-133 Rev. 2 (Section 4, Ex. 1) Generation of Symmetric Keys

Table 7: Vendor-Affirmed Algorithms

Non-Approved, Allowed Algorithms:

N/A for this module.

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

N/A for this module.

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Digital Signature	DigSig-SigGen	Digital Signature Generation		RSA SigGen (FIPS186-4): (A4111) ECDSA SigGen (FIPS186-4): (A4111) SHA2-256: (A4112) SHA2-512: (A4112) SHA2-224: (A4112) SHA2-384: (A4112)
KAS-SSC1	KAS-SSC	EC Diffie-Hellman key agreement primitive for shared secret computation	IG:IG D.F Scenario 2 path (1)	KAS-ECC CDH-Component SP800-56Ar3: (A4111)
Symmetric Cipher (Authenticated)	BC-Auth	Authenticated encryption and decryption		AES-CCM: (A4111) AES-CMAC: (A4111) AES-GCM: (A4111) AES-GMAC: (A4111) AES-ECB: (A4112)
Symmetric Cipher (Unauthenticated)	BC-UnAuth	Encryption and decryption		AES-CBC: (A4111) AES-CTR: (A4111) AES-ECB: (A4111, A4112, A4113)

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Type	Description	Properties	Algorithms
Random Number Generator	DRBG	Generate symmetric key (AES, HMAC), GCM IV		Hash DRBG: (A4111)
Message Digest Generation	SHA	Message digest generation		SHA2-224: (A4111) SHA2-256: (A4111) SHA2-384: (A4111) SHA2-512: (A4111)
Keyed Hash	MAC	Generate or verify MAC		HMAC-SHA2-224: (A4111) HMAC-SHA2-256: (A4111) HMAC-SHA2-384: (A4111) HMAC-SHA2-512: (A4111)
Key Derivation	KBKDF	Counter Mode KDF using 128-bit CMAC-AES		AES-CMAC: (A4112) KDF SP800-108: (A4111)
Symmetric Key Generation	CKG	Generate symmetric key (AES, HMAC)		CKG: () Hash DRBG: (A4111)
Key Wrap/Unwrap	BC-AuthDecrypt BC-AuthEncrypt	Provide key wrapping for keys		AES-KW: (A4111)

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

AES-GCM:

The module implements internal IV generation per FIPS 140-3 IG C.H scenario 2. The IV is generated using an approved DRBG (Hash_DRBG) internal to the module and the IV length is 96 bits.

AES-KW:

The module implements AES-KW mode, and the calling application can use it for key wrapping and unwrapping. AES-KW is CAVP tested per FIPS 140-3 IG D.G.

RSA:

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

The module implements RSA signature generation and the algorithm has been tested and validated with all implemented modulus lengths for which CAVP testing is available (2048, 3072, and 4096) per FIPS 140-3 IG C.F.

KAS-ECC Component:

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

Key Wrapping:

The module does not establish SSPs using an approved key transport scheme (KTS). However, it does offer approved authenticated algorithms that can be used by an external operator/application as part of an approved KTS.

2.8 RBG and Entropy

An external RBG entropy source is required to be supplied for the generation of the DRBG seed via the callback function. The entropy in the seeding material input to the module must provide at least as much entropy as the security strength of the module's employed DRBG (256 bits).

Per FIPS 140-3 IG 9.3.A, the caveat "No assurance of the minimum strength of generated SSPs (e.g., keys)" applies to this module.

Source Entropy Input for the module:

As the module implements various algorithms that use the underlying DRBG mechanism, a source entropy input (SEI) must be made available for the module operation. The module has a callback function for the SEI that needs to be initialized before operation. The callback function's signature is as follows:

```
SKB_Result ApplicationEntropySource(SKB_Byte* buffer, SKB_Size count)
```

The first parameter is a byte buffer in which to store the entropy input. The second argument is the byte count of the requested entropy input.

The function, when successful, shall store the exact requested count of bytes with at least 256 bits of entropy in the first parameter.

The return value specifies if the callback was successful and will return 0 in case of normal operation. If an error is encountered, for instance the entropy of 256 bits cannot be achieved, the return value will be -1. The module will enter error state if the callback is unsuccessful.

The entropy source callback will be used only for instantiating and re-seeding the module's internal SHA2-256 HASH_DRBG.

The SKB API call to set the callback function is as follows:

```
SKB_FIPS_SetEntropySource(ApplicationEntropySource)
```


Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Hash_DRBG Specification:

The module implements SHA2-256 hash DRBG compliant to SP 800-90Arev1 publication. The DRBG supports only maximum 256-bit security strength and has no prediction resistance capability. The source of entropy input for the DRBG is completely under application control and the callback for entropy source must be set beforehand to make the operation of DRBG possible.

DRBG Health Testing:

Implementation health testing is performed for following functions of DRBG:

- Instantiate
- Re-seed
- Generate

Health testing checks if the expected error codes are returned when invalid parameters or context is passed and if DRBG KAT tests pass. Health testing is done before the actual function is performed. If any health check fails, the given DRBG context (on which the call to DRBG service was made) is placed in an error state and no further operations can be performed on that DRBG instance until it has been re-instantiated.

The instantiate function health test is initiated on each instantiation and performs the following checks:

- Instantiate with invalid parameters (null context, already instantiated context, context is in error state, invalid nonce/personalization string)
- Instantiate with entropy source failure (when entropy source does not return 0 because of insufficient entropy)
- Instantiate KAT test variants of nonce/personalization string passed/omitted

The re-seed function health test is initiated on each re-seeding (whether explicit or within the automatic re-seed inside the generate function) and performs the following checks:

- Re-seed with invalid parameters (null context, uninstantiated context, context in error state)
- Re-seed with entropy source failure (when entropy source does not return 0 because of insufficient entropy)
- Instantiate then re-seed KAT test

The generate function health test is initiated on the first run of generate for given DRBG context and on each subsequent 2^{24} generation. The check interval can be adjusted via a function call. The health checks performed in the generate function are as follows:

- Generate with invalid parameters (null context, non-instantiated context, context in error state, null buffer for output requested bytes, requested byte count too large)
- Instantiate then generate KAT test
- Instantiate then generate with triggered re-seed KAT test

2.9 Key Generation

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

The module implements SP 800-90Arev1 compliant DRBG algorithm used within the module for the following purposes:

- Generation of the secret random number k during the ECDSA signature
- Generation of the 96-bit IV for AES-GMAC and AES-GCM encryption
- Generation of AES and HMAC keys at request

The module does not output intermediate key generation values.

Per Section 4 of the SP 800-133rev2, random bit strings required for the generation of cryptographic keys are obtained from the output of an approved random bit generator (Hash_DRBG A4111).

2.10 Key Establishment

Not applicable for this module.

2.11 Industry Protocols

Not applicable for this module.

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	API entry point data input stack parameters
N/A	Data Output	API entry point data output stack parameters
N/A	Control Input	API entry point and corresponding stack parameters
N/A	Status Output	API entry point return values and status stack parameters

Table 9: Ports and Interfaces

The logical interfaces specified above are C-language APIs. As a software module, control of the physical ports is outside of the module scope. When the module is performing self-tests, or is in an error state, all output on the logical data output interface is inhibited. The module does not support a control output interface. The Module is single-threaded and in error scenarios returns only an error value (no data output is returned).

4 Roles, Services, and Authentication

4.1 Authentication Methods

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

The module does not support any kind of authentication. The module is not allowed to be used with concurrent operators.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto Officer	Role	CO	None

Table 10: Roles

The module supports the Cryptographic Officer (CO) role and does not support a maintenance role or a bypass capability. The role of the Crypto Officer is assumed implicitly when installing or operating the services of the module.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show Status	API call SKB_FIPS_GetErrorStatus returns the state of the module	N/A	N/A	Module status - operational ('0') or in fail state ('1')	None	Crypto Officer
Self-Test	API call SKB_FIPS_ExecuteIntegrityCheck executes the integrity test of the module and API call SKB_FIPS_ExecureSelfTests executes the module's CASTs	N/A	N/A	SKB_FIPS_ExecuteIntegrityCheck returns '0' for success and non-zero value for failure. SKB_FIPS_ExecureSelfTests returns '0' for success and '-1' for failure	None	Crypto Officer
Instantiate	Instantiate the module engine	N/A	N/A	Engine instance	None	Crypto Officer
Symmetric cipher	Encrypt and decrypt data (including authenticated modes)	Successful completion of the service and global status	Plaintext/ciphertext, encryption/decryption key	Ciphertext/plaintext and mac or tag in case of authenticated encryption	Symmetric Cipher (Authenticated) Symmetric Cipher (Unauthenticated) Symmetric Key	Crypto Officer - EDK: R,E

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		indicator '0' returned after executing Show Status service			Generation	
Message Digest	Generate message digest	Successful completion of the service and global status indicator '0' returned after executing Show Status service	Data	Digest	Message Digest Generation	Crypto Officer
Digital Signature	Generate digital signature	Successful completion of the service and global status indica	Private key, data	Signature	Digital Signature	Crypto Officer - PK: R,E

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		tor '0' returned after executing Show Status service				
Keyed Hash	Generate or verify MAC	Successful completion of the service and global status indicator '0' returned after executing Show Status service	Key, data/signature	MAC or verification result	Keyed Hash	Crypto Officer - KHK: R,E
Cryptographic Key Generation	Generate symmetric key (AES, HMAC)	Successful completion of the service and global status indicator '0'	N/A	Generated key	Random Number Generator Symmetric Key Generation	Crypto Officer - EDK: G,W - KEDK: G,W - KDK: G,W - KHK: G,W - SEI: W

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		returned after executing Show Status service				
Key Agreement Primitive	EC Diffie-Hellman key agreement primitive for shared secret computation	Successful completion of the service and global status indicator '0' returned after executing Show Status service	Private and peer public key	Shared secret	KAS-ECC	Crypto Officer - PK: R,E - PUK: R,E - SS: G,R
Key Derivation	SP 800-108rev1 key derivation function in counter mode using CMAC-AES	Successful completion of the service and global status indicator '0' return	Key, data (label, context)	Derived keying material	Key Derivation	Crypto Officer - KDK: R,W,E - EDK: W - KEDK: W - KHK: W

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		ed after executing Show Status service				
Key Wrapping	Provide key wrapping for keys	Successful completion of the service and global status indicator '0' returned after executing Show Status service	wrapping key, key to-be-wrapped or wrapped key	Wrapped or unwrapped key	Key Wrap/Unwrap	Crypto Officer - KEDK: R,W,E - EDK: R,W - KDK: R,W - KHK: R,W - SS: R,W
Import	Converts an exported key (from a pre-formatted byte buffer) to an appropriate module key type object for use in module services. Key is loaded from inside the physical perimeter of the GPC.	Successful completion of the service and global status indicator '0' returned	Key (byte buffer)	Key	None	Crypto Officer - PK: R - PUK: R - EDK: R - KEDK: R - KDK: R - KHK: R

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		after executing Show Status service				
Export	Converts the key from a module key type object to a pre-formatted byte buffer (within the physical perimeter of the GPC)	Successful completion of the service and global status indicator '0' returned after executing Show Status service	Key	Key (byte buffer)	None	Crypto Officer - PK: W - PUK: W - EDK: W - KEDK: W - KDK: W - KHK: W
Release Resources (Zeroise)	Deallocates/Zeroise the memory containing a symmetric key, asymmetric private key, cryptographic service state	Successful completion of the service and global status indicator '0' returned after	Engine instance, key or function object	Return code '0'	None	Crypto Officer - PK: Z - PUK: Z - SS: Z - EDK: Z - KEDK: Z - KDK: Z - KHK: Z -

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
		executing Show Status service				DRBG_SEED: Z - DRBG_C: Z - DRBG_v: Z - GCM IV: Z
Show Version	API call SKB_Engine_GetInfo returns version and name of the module	N/A	N/A	Version: 5.38.1, Name: Zimperium zKeyBox white-box cryptography library	None	Crypto Officer

Table 11: Approved Services

4.4 Non-Approved Services

N/A for this module.

The module does not implement any non-approved services.

4.5 External Software/Firmware Loaded

Not applicable for this module.

5 Software/Firmware Security

5.1 Integrity Techniques

The zKeyBox shared library file is protected by a HMAC-SHA2-256 MAC. The MAC value is pre-calculated and embedded in the delivered module object. This value is calculated on the code/text and constant/read-only data sections of the binary file. The value is stored in a specifically marked place in the constant data section of the module. This place is not included in the message input for the digest algorithm; thus, it does not influence the digest value. At run-time, the integrity validation check is executed once the module library is being loaded into memory. The integrity check performs HMAC-SHA2-256 on the code and data sections of the module in volatile memory and compares it to the embedded value. If the integrity check fails, further operations of services in the module are denied. This check can also be initiated at any time during the operation of the module by calling the integrity checking API method (SKB_FIPS_ExecuteIntegrityCheck) directly.

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

By doing so, the operator of the module can reassure from time to time that the module in memory has not been altered. All temporary values generated during the integrity test are zeroized after the check has completed.

5.2 Initiate on Demand

The operator can initiate the integrity test on demand by calling API method SKB_FIPS_ExecuteIntegrityCheck.

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Modifiable

How Requirements are Satisfied:

The module operates in a modifiable operational environment as per FIPS 140-3 Level 1 specifications. The operational environments are designed to accept functional changes that may contain non-controlled software.

7 Physical Security

As a software module, the physical security requirements are not applicable.

8 Non-Invasive Security

The module does not claim any non-invasive attack mitigation.

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
RAM	Volatile Memory, no persistent storage	Dynamic

Table 12: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Input	Calling application	API input parameters	Plaintext	Manual	Electronic	

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Output	API output parameters	Calling application	Plaintext	Manual	Electronic	

Table 13: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Reboot	Rebooting the host platform	Rebooting the host platform removes all the data stored in the RAM	Can be initiated by Crypto Officer
API call	SKB_ZeroMemory() function call to overwrite temporary integrity test SSP with zeros	The function will overwrite the memory with zeros	Automatically invoked at the end of the integrity test

Table 14: SSP Zeroization Methods

The parameters are zeroized procedurally by rebooting the host platform.

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
PK	Private component of an RSA (k = 2048, 3072 or 4096) or ECDSA (P-224, P-256, P-384, P-521) key pair for signature generation or private component of an ECC key pair for key agreement primitive	RSA: 2048, 3072, 4096. ECDSA/ECC: 224, 256, 384, 521 - RSA: 112, 128, 152. ECDSA/ECC: 112, 128, 192, 256	Private - CSP			Digital Signature KAS-ECC
PUK	Public component of an ECC (P-224, P-256, P-384, P-521) key pair for key	224, 256, 384, 521 - 112, 128, 192, 256	Public - CSP			KAS-ECC

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
	agreement primitive					
SS	Shared Secret (Z)	224, 256, 384, 521 - 112, 128, 192, 256	Secret - CSP	KAS-ECC CDH-Component SP800-56Ar3 (A4111)		
EDK	Encryption/decryption keys for encryption, authenticated encryption and authentication	128,192, 256 - 128, 192, 256	Symmetric - CSP	Symmetric Key Generation		Symmetric Cipher (Authenticated) Symmetric Cipher (Unauthenticated) Keyed Hash
KEDK	Encryption / decryption keys for key wrapping	128,192, 256 - 128,192, 256	Symmetric - CSP	Symmetric Key Generation		Key Wrap/Unwrap
KDK	Key Derivation Key	128 - 128	Symmetric - CSP	Symmetric Key Generation		Key Derivation
KHK	Keyed hash key for HMAC SHA2 224/256/384/512 operations	512-4096 - 112-256	Authentication - CSP	Symmetric Key Generation		Keyed Hash
DRBG_SEED	Seed used for DRBG instantiation and reseed	128-256 - 128-256	Seed - CSP	Random Number Generator		Random Number Generator
DRBG_C	Hash_DRBG State element	128-256 - 128-256	DRBG State - CSP	Random Number Generator		Random Number Generator
SEI	Entropy input from an external source used to seed the Hash_DRBG	256 - 256	Entropy - CSP			Random Number Generator
DRBG_v	Hash_DRBG State element	128-256 - 128-256	DRBG State - CSP	Random Number Generator		Random Number Generator

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
GCM IV	Initialization Vector used by AES-GCM	96 - 96	IV - CSP	Symmetric Key Generation		Symmetric Cipher (Authenticated)

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
PK	Input	RAM:Plaintext	API call lifetime	Reboot	
PUK	Input	RAM:Plaintext	API call lifetime	Reboot	
SS	Output	RAM:Plaintext	API call lifetime	Reboot	
EDK	Input Output	RAM:Plaintext	API call lifetime	Reboot	
KEDK	Input Output	RAM:Plaintext	API call lifetime	Reboot	
KDK	Input Output	RAM:Plaintext	API call lifetime	Reboot	
KHK	Input Output	RAM:Plaintext	API call lifetime	Reboot	
DRBG_SEED		RAM:Plaintext	API call lifetime	Reboot	
DRBG_C		RAM:Plaintext	API call lifetime	Reboot	
SEI	Input	RAM:Plaintext	API call lifetime	Reboot	
DRBG_v		RAM:Plaintext	API call lifetime	Reboot	
GCM IV		RAM:Plaintext	API call lifetime	Reboot	

Table 16: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A4111)	HMAC-SHA2-256	HMAC integrity test over code and data sections of the module	SW/FW Integrity	0 for success and non-zero for failure	HMAC verification

Table 17: Pre-Operational Self-Tests

The cryptographic algorithm self-test for HMAC-SHA2-256 is performed prior to the integrity test.

10.2 Conditional Self-Tests

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC Encrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
AES-CBC Decrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Decrypt	Module load
AES-CCM Encrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
AES-CCM Decrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Decrypt	Module load
AES-CMAC Generate (A4111)	128, 192, 256 bit	KAT	CAST	0	Generate	Module load
AES-CMAC Verify (A4111)	128, 192, 256 bit	KAT	CAST	0	Verify	Module load
AES-CTR Encrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
AES-CTR Decrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Decrypt	Module load
AES-ECB Encrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
AES-ECB Decrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Decrypt	Module load
AES-GCM Encrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
AES-GCM Decrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Decrypt	Module load
AES-KW Encrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
AES-KW Decrypt (A4111)	128, 192, 256 bit	KAT	CAST	0	Decrypt	Module load
ECDSA SigGen (FIPS186-4) (A4111)	P-256	KAT	CAST	0	Signature Generation	Module load
Hash DRBG (A4111)	SHA2-256	KAT	CAST	0	Health Tests: Generate,	Module load

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
					Reseed, Instantiate	
HMAC-SHA2-256 (A4111)	HMAC-SHA2-256	KAT	CAST	0	Generate	Module load
HMAC-SHA2-512 (A4111)	HMAC-SHA2-512	KAT	CAST	0	Generate	Module load
KAS-ECC CDH-Component SP800-56Ar3 (A4111)	P-256	KAT	CAST	0	Shared Secret Computation	Module load
KDF SP800-108 (A4111)	128 bit	KAT	CAST	0	SP 800-108rev1 KAT for Counter Mode KDF	Module load
RSA SigGen (FIPS186-4) (A4111)	2048 bit	KAT	CAST	0	Signature Generation	Module load
SHA2-256 (A4111)	SHA2-256	KAT	CAST	0	SHA2-256	Module load
SHA2-512 (A4111)	SHA2-512	KAT	CAST	0	SHA2-512	Module load
AES-CMAC Generate (A4112)	128, 192, 256 bit	KAT	CAST	0	Generate	Module load
AES-CMAC Verify (A4112)	128, 192, 256 bit	KAT	CAST	0	Verify	Module load
AES-ECB Encrypt (A4112)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
SHA2-256 (A4112)	SHA2-256	KAT	CAST	0	SHA2-256	Module load
SHA2-512 (A4112)	SHA2-512	KAT	CAST	0	SHA2-512	Module load
AES-ECB Encrypt (A4113)	128, 192, 256 bit	KAT	CAST	0	Encrypt	Module load
AES-ECB Decrypt (A4113)	128, 192, 256 bit	KAT	CAST	0	Decrypt	Module load

Table 18: Conditional Self-Tests

When the module library is being loaded in memory by a calling application, all self-tests listed above, and a subsequent run-time integrity check are performed. In case of failure, the module's error status, which indicates whether the module is operable, is set to either 1 or 2. The error status value can be retrieved via SKB_FIPS_GetErrorStatus function. In case when the value of the error status is not 0, all the cryptographic services of the module are denied.

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4111)	HMAC integrity test over code and data sections of the module	SW/FW Integrity	On demand or module load	On demand: SKB_FIPS_ExecuteIntegrityCheck(), module load: Initialization

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC Encrypt (A4111)	KAT	CAST	On demand	Module load
AES-CBC Decrypt (A4111)	KAT	CAST	On demand	Module load
AES-CCM Encrypt (A4111)	KAT	CAST	On demand	Module load
AES-CCM Decrypt (A4111)	KAT	CAST	On demand	Module load
AES-CMAC Generate (A4111)	KAT	CAST	On demand	Module load
AES-CMAC Verify (A4111)	KAT	CAST	On demand	Module load
AES-CTR Encrypt (A4111)	KAT	CAST	On demand	Module load
AES-CTR Decrypt (A4111)	KAT	CAST	On demand	Module load
AES-ECB Encrypt (A4111)	KAT	CAST	On demand	Module load
AES-ECB Decrypt (A4111)	KAT	CAST	On demand	Module load
AES-GCM Encrypt (A4111)	KAT	CAST	On demand	Module load
AES-GCM Decrypt (A4111)	KAT	CAST	On demand	Module load
AES-KW Encrypt (A4111)	KAT	CAST	On demand	Module load
AES-KW Decrypt (A4111)	KAT	CAST	On demand	Module load
ECDSA SigGen (FIPS186-4) (A4111)	KAT	CAST	On demand	Module load
Hash DRBG (A4111)	KAT	CAST	On demand	Module load

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A4111)	KAT	CAST	On demand	Module load
HMAC-SHA2-512 (A4111)	KAT	CAST	On demand	Module load
KAS-ECC CDH-Component SP800-56Ar3 (A4111)	KAT	CAST	On demand	Module load
KDF SP800-108 (A4111)	KAT	CAST	On demand	Module load
RSA SigGen (FIPS186-4) (A4111)	KAT	CAST	On demand	Module load
SHA2-256 (A4111)	KAT	CAST	On demand	Module load
SHA2-512 (A4111)	KAT	CAST	On demand	Module load
AES-CMAC Generate (A4112)	KAT	CAST	On demand	Module load
AES-CMAC Verify (A4112)	KAT	CAST	On demand	Module load
AES-ECB Encrypt (A4112)	KAT	CAST	On demand	Module load
SHA2-256 (A4112)	KAT	CAST	On demand	Module load
SHA2-512 (A4112)	KAT	CAST	On demand	Module load
AES-ECB Encrypt (A4113)	KAT	CAST	On demand	Module load
AES-ECB Decrypt (A4113)	KAT	CAST	On demand	Module load

Table 20: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
Error state	Entered when pre-operational self-test or conditional self-test fails	Pre-operational self-test failure Conditional self-test failure	Module reload	SKB_FIPS_GetErrorStatus returns 1

Table 21: Error States

If any component of the self-test or the integrity check fails, the functions mentioned in Section 10.5 return the error value (-1 in case of KAT failure) and the module's error status is set accordingly to prevent subsequent invocation of any cryptographic function calls.

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

10.5 Operator Initiation of Self-Tests

The self-tests may also be performed on-demand by calling SKB_FIPS_ExecuteSelfTests function. The same is true for the integrity checking function SKB_FIPS_ExecuteIntegrityCheck. Upon successful execution, these functions return 0 return value.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

Below are the installation, initialization and startup procedures:

- Unzip the module's package archive in a convenient location.
- Add the module's shared library (framework for iOS) to the linking step of user application's project. The module's files are located in the following directories of the package root:
    - "lib/google-android/arm64-v8a" for the Android version of the library
    - "lib/apple/iphoneos" for the iOS version of the library

In the application's source code:

- Include the SkbSecureKeyBox.h" and "SkbSecureKeyBoxFips.h" header files from the "Include" directory within the package's root.
- Set the entropy source as described in Section 2.8 "RGB and Entropy" and acquire the module's engine instance.
- Use the module's services as needed.

11.2 Administrator Guidance

Any additional guidance documentation, if required, can be obtained by contacting Zimperium, Inc.

11.3 Non-Administrator Guidance

Not applicable for this module.

11.4 Design and Rules

Rules of operation enforced by the cryptographic module to implement the security requirements of this FIPS 140-3 Level 1 module are as follows:

- The operating system shall be restricted to a single operator mode of operation (i.e., concurrent operators are explicitly excluded).

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.

- The module shall be operated in single user/operator mode. The external application that makes calls to the cryptographic module is the single user of the cryptographic module, even when the application is serving multiple clients.
- Unauthorized reading, writing, or modification of the address space of the module is prohibited.
- The referencing application accessing the module runs in a separate virtual address space with a separate copy of the executable code.
- The operating system is responsible for multi-tasking operations so that other processes cannot access the address space of the process containing the module.
- The operator (calling application) shall use appropriate entropy sources (entropy source callback in the module) for generating the seeding material for the approved DRBG of the module. The entropy in the seeding material input to the module must be at least as much as the security strength of the module's employed DRBG (256 bits).

11.5 Additional Information

Subversion (SVN), Jenkins, and Zimperium proprietary extensive SKB test suite and build scripts are used to provide testing and configuration management for the module and related documentation. These solutions provide access control, versioning, testing, and logging.

The testing of the Module includes:

- Unit testing (proprietary test suite)
- Memory sanitization (GCC)
- Memory leak detection (Valgrind)

12 Mitigation of Other Attacks

The module does not claim any attack mitigation.

Copyright © 2026 Zimperium, Inc.
This non-proprietary Security Policy document may be freely reproduced and distributed in its entirety without modification.