Ideem, Inc.

Ideem ZSM Cryptographic Module

Software Version: 1.0

FIPS 140-3 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1.0

Prepared for:

Ideem, Inc.
800 W. 47th Street
Kansas City, MO 64112
United States of America

Phone: +1 816 280 4456
www.useideem.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

Abstract

This is a non-proprietary Cryptographic Module Security Policy for the Ideem ZSM Cryptographic Module (version: 1.0) from Ideem, Inc. (Ideem). This Security Policy describes how the Ideem ZSM 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 National Institute of Standards and Technology (NIST) and the Canadian Centre for Cyber Security (CCCS) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/cmvp.

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 Ideem ZSM Cryptographic Module is referred to in this document as Ideem Cryptographic Module or the module.

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 Ideem website www.useideem.com contains information on the full line of services and solutions from Ideem.

- 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.

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.

Table of Contents

List of Tables

List of Figures

1. General

1.1 Overview

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

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

1.2 Security Levels

The Ideem ZSM Cryptographic Module is validated at the FIPS 140-3 section levels shown 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	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	1
	Overall Level	1

Table 1: Security Levels

The module has an overall security level of 1.

2. Cryptographic Module Specification

2.1 Description

2.1.1 Purpose and Use

Ideem Cryptographic Module is able to protect all types of standard cryptographic keys for all purposes, including encryption/decryption, digital signing, and authentication. Ideem's technology for securing keys secure multiparty computation (MPC) is fully transparent to the calling application.

2.1.2 Module Type

The Ideem ZSM Cryptographic Module 1.0 is a Software module.

2.1.3 Module Embodiment

The Ideem ZSM Cryptographic Module has a MultiChipStand embodiment.

The module is designed to utilize the following processor algorithm acceleration (PAA) instructions sets for its AES and SHA implementations:

- AES-NI instruction set, when executing on the RHEL 9.2 operational environment
- Neon instruction set, when executing on the iOS 16.5 or Android 13 operational environments.

The module was tested and found to be compliant with FIPS 140-3 requirements on the environments listed in section 2.2.4 of this Security Policy.

2.1.4 Module Characteristics

The module does not have any additional characteristics.

2.1.5 Cryptographic Boundary

As a software cryptographic module, the module takes on the physical characteristics of the host platform. The physical perimeter of the cryptographic module is defined by each host device on which the module is installed.

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 module is entirely contained within the physical perimeter.

The module's cryptographic boundary consists of all functionalities contained within the module's compiled source code. The module's software component comprises 3 shared library files and 3 digest files for testing integrity. All hardware and software components are contained within the host platform's physical enclosure.

The module's cryptographic boundary consists of all functionalities contained within the module's compiled source code.
On the RedHat and Android operating systems the module is comprised of the following binary files:

- libcrypto.so (cryptographic primitives library file)
- libsecurikey.so (cryptographic primitives library file)
- libssl.so (TLS protocol library file)
- libcrypto.hmac (an HMAC digest file for libcrypto integrity check)
- libsecurikey.hmac (an HMAC digest file for libsecurikey integrity check)
- libssl.hmac (an HMAC digest file for libssl integrity check)

and on the iOS operating system:

- libcrypto.dylib (cryptographic primitives library file)
- libsecurikey.dylib (cryptographic primitives library file)
- libssl.dylib (TLS protocol library file)
- libcrypto.hmac (an HMAC digest file for libcrypto integrity check)
- libsecurikey.hmac (an HMAC digest file for libsecurikey integrity check)
- libssl.hmac (an HMAC digest file for libssl integrity check)

The module is entirely contained within the physical perimeter.

2.1.6 Tested Operational Environment's Physical Perimeter (TOEPP)

Figure 1 illustrates a block diagram of a typical GPC and the module's physical perimeter.

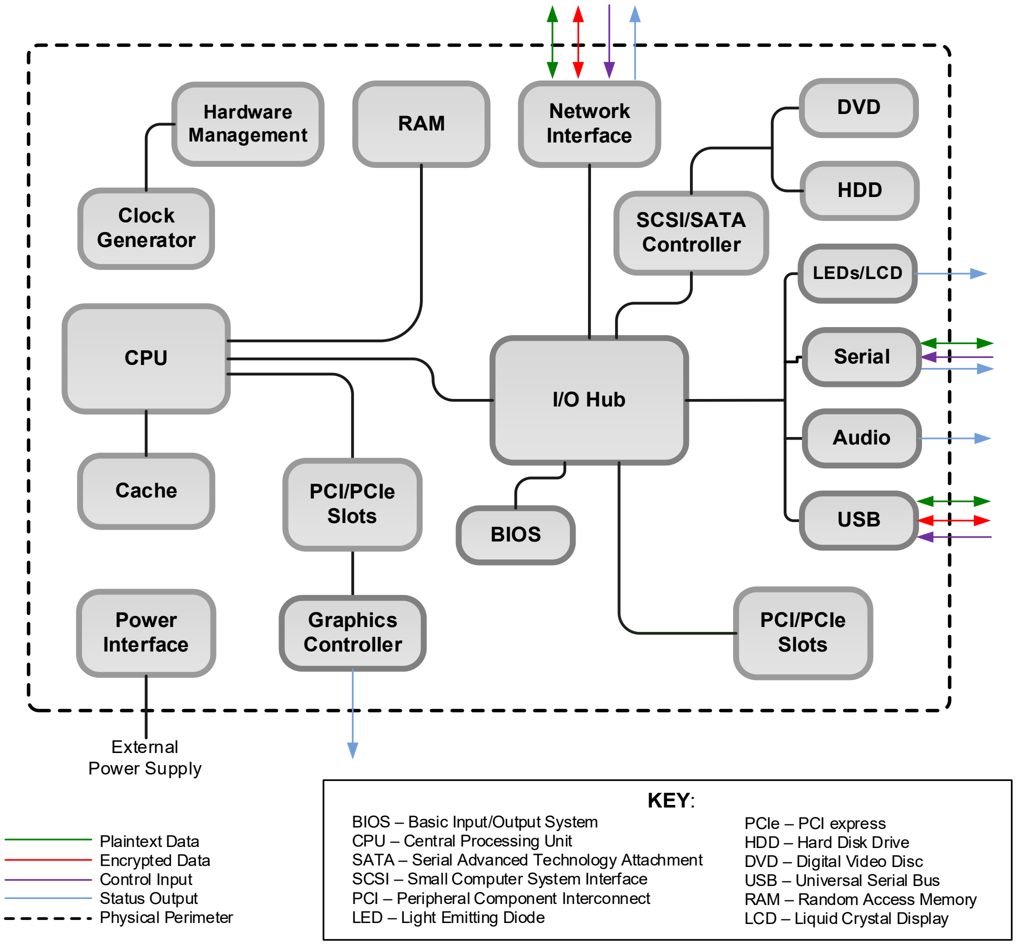

Figure 1. GPC Block Diagram

Figure 2 below shows the logical block diagram of the module executing in memory and its interactions with surrounding software components, as well as the module's physical perimeter and logical cryptographic boundary.

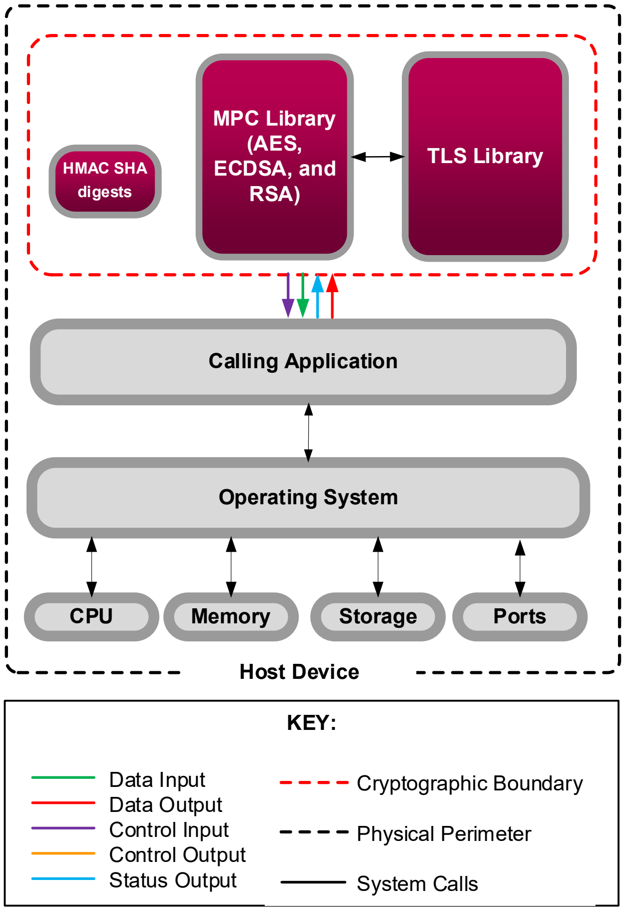

Figure 2. Module Block Diagram (with Cryptographic Boundaries)

2.2 Tested and Vendor Affirmed Module Version and Identification

2.2.1 Tested Module Identification – Hardware

This section is only applicable to hardware modules.

N/A for this module.

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

The table below lists the executable code sets of the module. The module count is 3.

Package or File Name	Software/ Firmware Version	Features	Integrity Test
RedHat: libcrypto.so, libsecurikey.so, libssl.so, libcrypto.hmac, libsecurikey.hmac, libssl.hmac	1.0		Yes
Android: libcrypto.so, libsecurikey.so, libssl.so, libcrypto.hmac, libsecurikey.hmac, libssl.hmac	1.0		Yes
iOS: libcrypto.dylib, libsecurikey.dylib, libssl.dylib, libcrypto.hmac, libsecurikey.hmac, libssl.hmac	1.0		Yes

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.

N/A for this module.

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)
RedHat 9.2	Dell T5610	Intel Xeon CPU E5-2609 v2	Yes		1.0
RedHat 9.2	Dell T5610	Intel Xeon CPU E5-2609 v2	No		1.0
iOS 16.5	iPhone 14	Apple A15 Bionic (ARMv8)	Yes		1.0
iOS 16.5	iPhone 14	Apple A15 Bionic (ARMv8)	No		1.0
Android 13	Pixel 6	Octa-core (2x2.80 GHz Cortex-X1 & 2x2.25 GHz Cortex-A76 & 4x1.80 GHz Cortex-A55)	Yes		1.0
Android 13	Pixel 6	Octa-core (2x2.80 GHz Cortex-X1 & 2x2.25 GHz Cortex-A76 & 4x1.80 GHz Cortex-A55)	No		1.0

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

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

The vendor affirms the module's continued validation compliance when operating on the platforms listed below. The table below also lists operational environments that support the mixed configuration.

Operating System	Hardware Platform
RedHat 7 and above	Any x86-based platform

Operating System	Hardware Platform
CentOS 7 and above	Any x86-based platform
Ubuntu 16 and above	Any x86-based platform
iOS 12 and above	iPhone 8 and newer
Android 11 and above	Pixel 4a and newer
RedHat 7 and above and iOS 12 and above	Any x86-based platform and iPhone 8 and newer (mixed configuration)
CentOS 7 and above and iOS 12 and above	Any x86-based platform and iPhone 8 and newer (mixed configuration)
Ubuntu 16 and above and iOS 12 and above	Any x86-based platform and iPhone 8 and newer (mixed configuration)
RedHat 7 and above and Android 11 and above	Any x86-based platform and Pixel 4a and newer (mixed configuration)
CentOS 7 and above and Android 11 and above	Any x86-based platform and Pixel 4a and newer (mixed configuration)
Ubuntu 16 and above and Android 11 and above	Any x86-based platform and Pixel 4a and newer (mixed configuration)

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

The mixed configuration operating environments refer to a distributed system where two instances of the cryptographic module are configured to execute the multi-party services in communication with each other, via the calling Application.

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

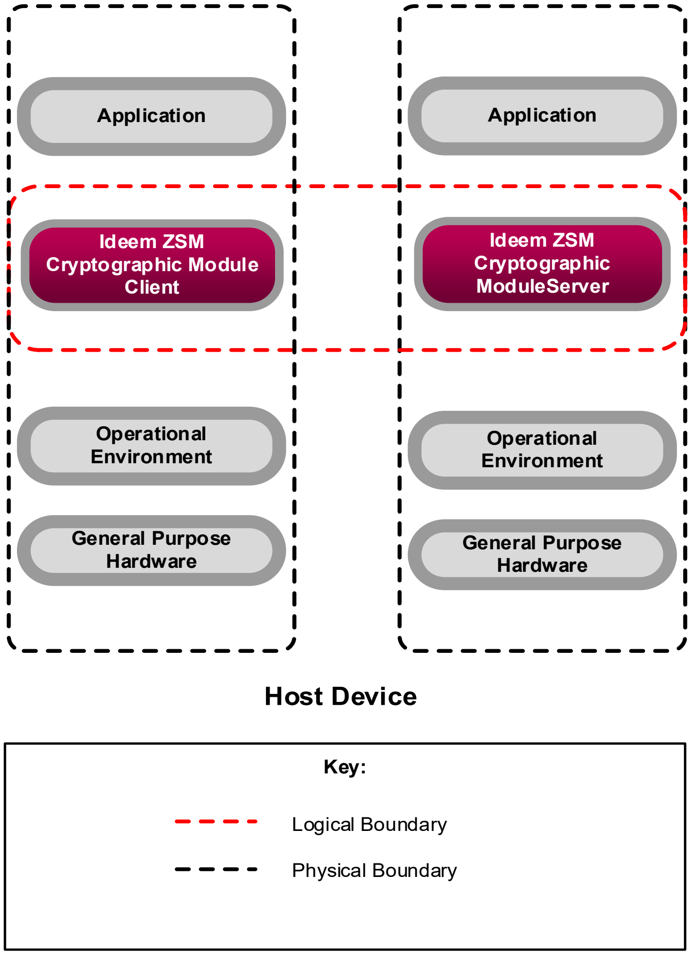

Figure 3 - Logical Diagram of the Cryptographic Module Running on Different Machines

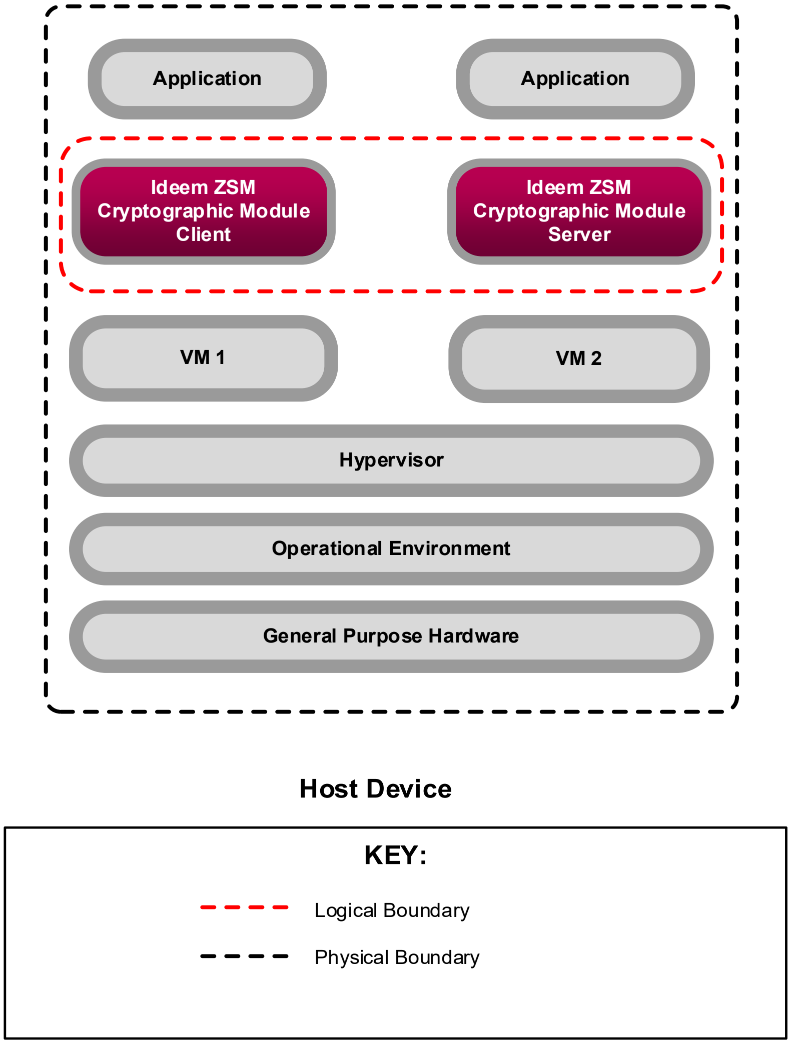

Figure 4 - Logical Diagram of the Cryptographic Module Running on Different Virtual Machines in the Same Hypervisor

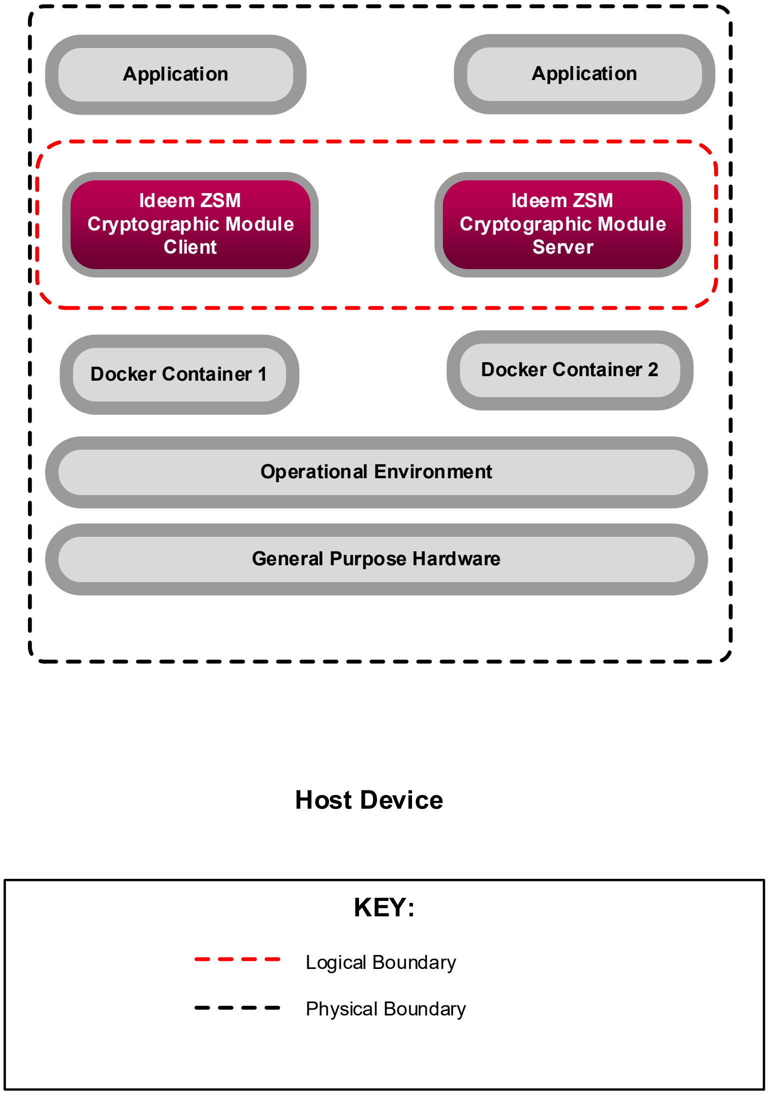

Figure 5 - Logical Diagram of the Cryptographic Module Running on Different Docker Containers

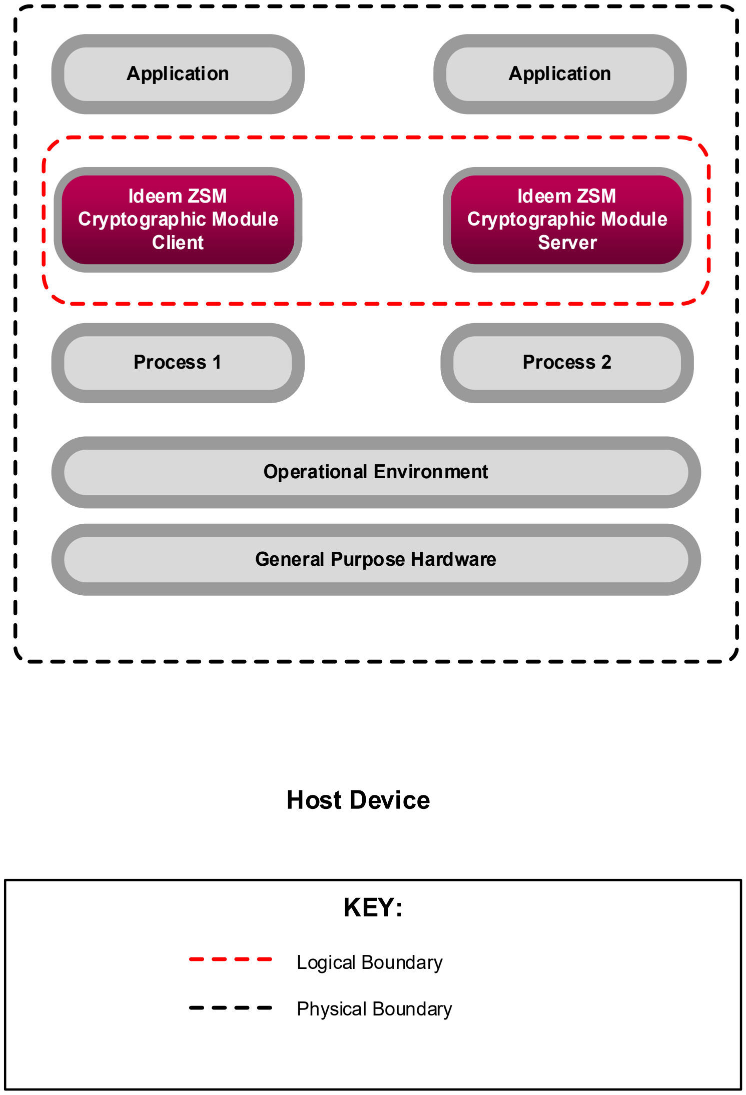

Figure 6 - Logical Diagram of the Cryptographic Module Running on Different Processes

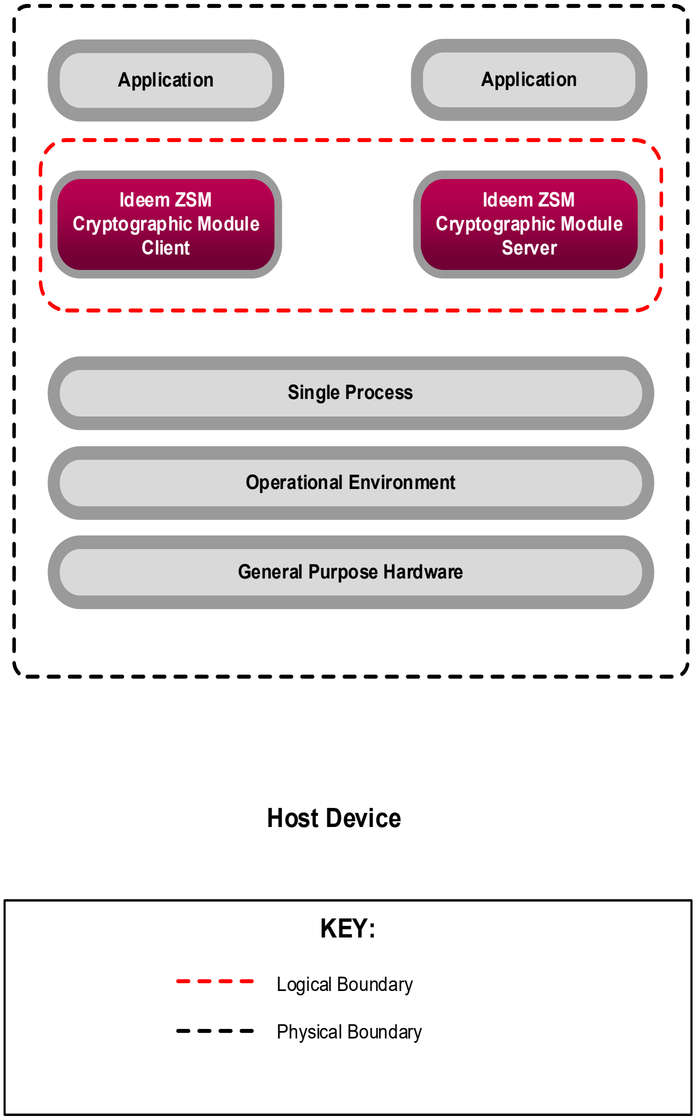

Figure 7 - Logical Diagram of the Cryptographic Module Running on a Single Process

The cryptographic module maintains validation compliance when operating on any general-purpose computer (GPC) provided that the GPC for the software module uses any single-user operating system/mode specified on the validation certificate, or another compatible single-user operating system.

The module also maintains compliance when operating in a virtual environment provided by any of the following supported hypervisors:

- VMware ESXi 6 and above

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

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

By design, the module only supports operation in the Approved mode.

Mode Name	Description	Type	Status Indicator
Approved	Mode allows the use of cryptographic operations	Approved	

Table 5: Modes List and Description

2.5 Algorithms

2.5.1 Approved Algorithms

The module employs cryptographic algorithm implementations from the following sources:

- Ideem ZSM Multi-party Cryptographic Library (Cert. A5055)
- Ideem ZSM Single-party Cryptographic Library (Cert. A5056)

The module implements the Approved algorithms listed below.

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CCM	A5056	Key Length - 128, 192, 256	SP 800-38C
AES-CFB1	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A

Algorithm	CAVP Cert	Properties	Reference
AES-CFB128	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CFB8	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CMAC	A5056	Direction - Generation, Verification Key Length - 128, 192, 256	SP 800-38B
AES-CTR	A5055	Direction - Decrypt, Encrypt Key Length - 128	SP 800-38A
AES-CTR	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-ECB	A5055	Direction - Encrypt Key Length - 128	SP 800-38A
AES-ECB	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-GCM	A5056	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-GMAC	A5056	Direction - Decrypt, Encrypt IV Generation - Internal IV Generation Mode - 8.2.1 Key Length - 128, 192, 256	SP 800-38D
AES-KW	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-KWP	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38F
AES-OFB	A5056	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A5056	Direction - Decrypt, Encrypt Key Length - 128, 256	SP 800-38E
Counter DRBG	A5056	Prediction Resistance - No, Yes Mode - AES-128, AES-192, AES-256 Derivation Function Enabled - Yes	SP 800-90A Rev. 1
ECDSA KeyGen (FIPS186-4)	A5055	Curve - P-256 Secret Generation Mode - Testing Candidates	FIPS 186-4
ECDSA KeyGen (FIPS186-5)	A5056	Curve - P-224, P-256, P-384, P-521 Secret Generation Mode - testing candidates	FIPS 186-5
ECDSA KeyVer (FIPS186-5)	A5056	Curve - P-224, P-256, P-384, P-521	FIPS 186-5
ECDSA SigGen (FIPS186-4)	A5055	Curve - P-256 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	FIPS 186-4
ECDSA SigGen (FIPS186-5)	A5056	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
ECDSA SigVer (FIPS186-5)	A5056	Curve - P-224, P-256, P-384, P-521 Hash Algorithm - SHA2-224, SHA2-256, SHA2-384, SHA2-512	FIPS 186-5
HMAC-SHA-1	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-224	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-256	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-384	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/224	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA2-512/256	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-224	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1

Algorithm	CAVP Cert	Properties	Reference
HMAC-SHA3-256	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-384	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
HMAC-SHA3-512	A5056	Key Length - Key Length: 8-524288 Increment 8	FIPS 198-1
KAS-ECC-SSC Sp800-56Ar3	A5056	Domain Parameter Generation Methods - P-224, P-256, P-384, P-521 Scheme - ephemeralUnified - KAS Role - initiator, responder	SP 800-56A Rev. 3
KDA HKDF SP800-56Cr2	A5056	Derived Key Length - 2048 Shared Secret Length - Shared Secret Length: 224-3968 Increment 8 HMAC Algorithm - SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA2-512/224, SHA2-512/256, SHA3-224, SHA3-256, SHA3-384, SHA3-512	SP 800-56C Rev. 2
KDF TLS (CVL)	A5056	TLS Version - v1.2 Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1
PBKDF	A5056	Iteration Count - Iteration Count: 10-10000 Increment 1 Password Length - Password Length: 8-128 Increment 1	SP 800-132
RSA KeyGen (FIPS186-5)	A5055	Key Generation Mode - probable Modulo - 2048, 3072, 4096 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA KeyGen (FIPS186-5)	A5056	Key Generation Mode - probable Modulo - 2048, 3072, 4096 Primality Tests - 2powSecStr Private Key Format - standard	FIPS 186-5
RSA SigGen (FIPS186-5)	A5055	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigGen (FIPS186-5)	A5056	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
RSA SigVer (FIPS186-5)	A5056	Modulo - 2048, 3072, 4096 Signature Type - pkcs1v1.5, pss	FIPS 186-5
SHA-1	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-224	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-256	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-384	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512/224	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA2-512/256	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 180-4
SHA3-224	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHA3-256	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHA3-384	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHA3-512	A5056	Message Length - Message Length: 0-65528 Increment 8	FIPS 202
SHAKE-128	A5056	Output Length - Output Length: 16-1024 Increment 8	FIPS 202
SHAKE-256	A5056	Output Length - Output Length: 16-1024 Increment 8	FIPS 202
TLS v1.2 KDF RFC7627 (CVL)	A5056	Hash Algorithm - SHA2-256, SHA2-384, SHA2-512	SP 800-135 Rev. 1

Table 6: Approved Algorithms

2.5.2 Vendor Affirmed Algorithms

The vendor affirms the following cryptographic security methods:

Name	Properties	Implementation	Reference
CKG		Ideem ZSM Single-party Cryptographic Library	SP 800-133 Rev. 2

Table 7: Vendor-Affirmed Algorithms

2.5.3 Non-Approved, Allowed Algorithms

The module does not offer any non-Approved algorithms allowed 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 offer any non-Approved algorithms allowed in the Approved mode of operation with no security claimed.

N/A for this module.

2.5.5 Non-Approved, Not Allowed Algorithms

The module does not offer non-Approved algorithms not allowed 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 Multi-party Symmetric Encryption/Decryption	BC-UnAuth	Block cipher unauthenticated		AES-CTR AES-ECB
ECDSA for Multi-party Asymmetric Key Pair Generation	AsymKeyPair-KeyGen	Asymmetric key-pair generation		ECDSA KeyGen (FIPS186-4)
ECDSA for Multi-party Digital Signature Generation	DigSig-SigGen	Digital signature generation		ECDSA SigGen (FIPS186-4)
RSA for Multi-party Asymmetric Key Pair Generation	AsymKeyPair-KeyGen	Asymmetric key-pair generation		RSA KeyGen (FIPS186-5)
RSA for Multi-party Digital Signature Generation	DigSig-SigGen	Digital signature generation		RSA SigGen (FIPS186-5)
AES for Symmetric Encryption/Decryption	BC-UnAuth	Block cipher unauthenticated		AES-CBC AES-CFB1 AES-CFB8 AES-CFB128 AES-CTR AES-ECB AES-OFB
AES-CMAC for Message Authentication	MAC	Message Authentication		AES-CMAC

Name	Type	Description	Properties	Algorithms
AES-GMAC for Message Authentication	MAC	Message Authentication		AES-GMAC
AES-CCM for Authenticated Symmetric Encryption/Decryption	BC-Auth	Block cipher authenticated		AES-CCM
AES-GCM for Authenticated Symmetric Encryption/Decryption	BC-Auth	Block cipher authenticated		AES-GCM
AES-XTS for Symmetric Encryption/Decryption	BC-UnAuth	Block cipher unauthenticated		AES-XTS Testing Revision 2.0
DRBG	DRBG	Deterministic random bit generator		Counter DRBG
ECDSA for Key Generation	AsymKeyPair-KeyGen	Asymmetric key-pair generation		ECDSA KeyGen (FIPS186-5) Counter DRBG
ECDSA for Key Verification	AsymKeyPair-KeyVer	Asymmetric key-pair verification		ECDSA KeyVer (FIPS186-5)
ECDSA for Digital Signature Generation	DigSig-SigGen	Digital signature generation		ECDSA SigGen (FIPS186-5) SHA2-224 SHA2-256 SHA2-384 SHA2-512 Counter DRBG
ECDSA for Digital Signature Verification	DigSig-SigVer	Digital signature verification		ECDSA SigVer (FIPS186-5) SHA2-224 SHA2-256 SHA2-384 SHA2-512
HMAC for Message Authentication	MAC	Message authentication		HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-512/224 HMAC-SHA2-512/256 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512 SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA2-512/224 SHA2-512/256 SHA3-224 SHA3-256 SHA3-384 SHA3-512

Name	Type	Description	Properties	Algorithms
ECDH Shared Secret Computation	KAS-SSC	Shared secret computation		KAS-ECC-SSC Sp800-56Ar3 ECDSA KeyGen (FIPS186-5) ECDSA KeyVer (FIPS186-5) Counter DRBG
AES for Key Wrapping/Unwrapping	KTS-Wrap	Key Wrap		AES-CBC AES-CFB8 AES-CFB1 AES-CFB128 AES-CTR AES-ECB AES-OFB AES-CMAC AES-GMAC AES-CCM AES-GCM AES-KW AES-KWP HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-512/224 HMAC-SHA2-512/256 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512 SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA2-512/224 SHA2-512/256 SHA3-224 SHA3-256 SHA3-384 SHA3-512

Name	Type	Description	Properties	Algorithms
HKDF for Key Derivation	KAS-56CKDF	HMAC-based Extract-and-Expand Key Derivation Function		KDA HKDF SP800-56Cr2 HMAC-SHA-1 HMAC-SHA2-224 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 HMAC-SHA2-512/224 HMAC-SHA2-512/256 HMAC-SHA3-224 HMAC-SHA3-256 HMAC-SHA3-384 HMAC-SHA3-512 SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA2-512/224 SHA2-512/256 SHA3-224 SHA3-256 SHA3-384 SHA3-512
PBKDF for Key Derivation	PBKDF	Password-based key derivation		PBKDF SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512
RSA for Key Generation	AsymKeyPair-KeyGen	Key generation		RSA KeyGen (FIPS186-5) Counter DRBG
RSA for Signature Generation	DigSig-SigGen	Signature generation		RSA SigGen (FIPS186-5) Counter DRBG SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA2-512/224 SHA2-512/256 SHA3-224 SHA3-256 SHA3-384 SHA3-512

Name	Type	Description	Properties	Algorithms
RSA for Signature Verification	DigSig-SigVer	Signature verification		RSA SigVer (FIPS186-5) SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA2-512/224 SHA2-512/256 SHA3-224 SHA3-256 SHA3-384 SHA3-512
SHA for Message Digest	SHA	Message Digest		SHA-1 SHA2-224 SHA2-256 SHA2-384 SHA2-512 SHA2-512/224 SHA2-512/256
SHA3 for Message Digest	SHA	Message Digest		SHA3-224 SHA3-256 SHA3-384 SHA3-512
SHAKE for Extendable Output Function	XOF	Extendable output function		SHAKE-128 SHAKE-256
TLS v1.2 Key Agreement	KAS-Full	Key agreement		KAS-ECC-SSC Sp800-56Ar3 ECDSA KeyGen (FIPS186-5) ECDSA KeyVer (FIPS186-5) SHA2-256 SHA2-384 SHA2-512 Counter DRBG TLS v1.2 KDF RFC7627 KDF TLS
TLS v1.2 Authentication	DigSig-SigVer	TLS v1.2 signature verification		RSA SigVer (FIPS186-5) Modulo: 2048 Hash Algorithm: SHA2-256 SHA2-256
TLS v1.2 Data Encryption/Decryption	BC-Auth BC-UnAuth	TLS v1.2 encryption and decryption		AES-CBC Key size: 128, 256 Direction: Encrypt, Decrypt AES-GCM Key Length: 128, 256 IV Generation: Internal IV Generation Mode: 8.2.1 Direction: Decrypt, Encrypt

Name	Type	Description	Properties	Algorithms
TLS v1.2 Key Derivation	KAS-135KDF	TLS v1.2 key derivation		KDF TLS TLS v1.2 KDF RFC7627 SHA2-256 SHA2-384 SHA2-512

Table 8: Security Function Implementations

2.7 Algorithm Specific Information

The information below provides algorithm information of references to specifications.

- AES GCM IV: As per *I.G. C.H Key/IV Pair Uniqueness Requirements* from *SP 800-38D*, AES GCM IV implements scenario 1.

- AES GCM encryption is used in the context of the TLS protocol versions 1.2. 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 of *NIST SP 800-52rev2*. Per scenario 1 in *FIPS 140-3 IG* C.H, the mechanism for IV generation 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.

  In the event that 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.

- RSA: As per *FIPS 186-5 Appendix A.1*, RSA KeyGen implements the method discussed in A.1.3 Generation of Random Primes that are Probably Prime.

- ECDSA: As per *FIPS 186-5 Appendix A.2*, ECDSA KeyGen implements the method discussed in A.2.2 ECDSA Key Pair Generation by Rejection Sampling. As per section C.K of the Implementation Guidance, Ideem ZSM Multi-party Cryptographic Library is complaint with FIPS 186-5, as the FIPS 186-4 CAVP tests for ECDSA KeyGen, KeyVer, SigGen, and SigVer are mathematically identical to the FIPS 186-5 CAVP tests.

- PBKDF2: As per SP 800-132 section 5.4, PBKDF2 implements 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 $1/62^{20} = 7.044^{-35}$, 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 RBG and Entropy

The module does not have any entropy certificates.

N/A for this module.

The calling application provides a minimum number of 256 bits of entropy. The calling application and its entropy source are outside the module cryptographic boundary. The calling application shall use entropy sources that meet the security strength required for the Counter DRBG as shown in Table 3 of SP 800-90A Rev. 1. This entropy shall be supplied by means of a callback function. The callback function must return an error if the minimum entropy strength cannot be met.

N/A for this module.

2.9 Key Generation

Please refer to Table 8: Security Function Implementations for specification of the module's key generation methods.

2.10 Key Establishment

2.10.1 Key Agreement Information

Please refer to Table 8: Security Function Implementations for specification of the module's key agreement methods.

2.10.2 Key Transport Information

Please refer to Table 8: Security Function Implementations for specification of the module's transport methods.

2.11 Industry Protocols

The module implements the following industry protocol:

- TLS v1.2[1]

[1] No parts of the TLS protocol, other than the KDF, have been tested by the CAVP or CMVP.

3. Cryptographic Module Interfaces

3.1 Ports and Interfaces

The module supports the following logical interfaces:

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

The module does not support a “control output” interface.

As a software library, the cryptographic module has no direct access to any of the host platform’s physical ports; it communicates only to the calling application via its well-defined API. The table below contains a mapping of the physical and logical interfaces of the module.

Physical Port	Logical Interface(s)	Data That Passes
Physical data input port(s) of the tested platforms	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.
Physical data output port(s) of the tested platforms	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).
Physical control input port(s) of the tested platforms	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.
Physical status output port(s) of the tested platforms	Status Output	Logical interface is defined as API call return values. This includes status information regarding the module or invoked service/operation.

Table 9: Ports and Interfaces

4. Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support authentication mechanisms; operators implicitly assume an authorized role (or set of roles) based on the service selected.

N/A for this module.

4.2 Roles

The module supports a Crypto Officer (CO) that authorized operators can assume. The CO role performs cryptographic initialization or management functions and general security services.
The module also supports the following role:

- User – The User role performs general security services, including cryptographic operations and other approved security functions.

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

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

4.3 Approved Services

Descriptions of the services available are provided in the tables in this section.

This module is a software library that provides cryptographic functionality to calling applications. As such, the security functions provided by the module are considered the module's security services. Indicators for Approved services (in the case of this module, those security functions with algorithm validation certificates and all required self-tests) are provided via API return value.

When invoking a security function, the calling application provides inputs via an internal structure, or "context". Upon each service invocation, the module will determine if the invoked security function is an Approved service. To access the resulting value, the calling application must pass the finalized context to the indicator API associated with that security function (note the indicator check must be performed prior to any context cleanup is performed). The indicator API will return "1" to indicate the usage of an Approved 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.

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show status	Return FIPS mode status	N/A	API call parameters	Current operational status	None	Unauthenticated
Perform self-tests on-demand	Perform pre-operational self-tests	API return value	API call parameters	Status	None	Unauthenticated

Zeroize	Zeroize and de-allocate memory containing sensitive data	N/A	API call parameters	None	None	Crypto Officer - MPC AES Key: Z - MPC ECDSA Public key: Z - MPC ECDSA Private key: Z - MPC RSA public key: Z - MPC RSA private key: Z - AES key: Z - AES CCM key: Z - AES GCM key: Z - AES GCM IV: Z - AES XTS key: Z - AES CMAC key: Z - AES GMAC key: Z - ECDH private component: Z - ECDH public component: Z - DRBG entropy input: Z - DRBG seed: Z - DRBG 'V' value: Z - DRBG 'Key' value: Z - ECDSA private key: Z - ECDSA public key: Z - HMAC key: Z - RSA private key: Z - RSA public key: Z - TLS extended pre-master secret: Z - TLS master secret: Z - TLS Session Key: Z - TLS Authentication Key (HMAC key): Z - TLS Server Authentication Key: Z - HKDF Derived key: Z - PBKDF Derived key: Z - Password: Z

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Show versioning information	Return module versioning information	N/A	API call parameters	Module name, version	None	Unauthenticated
Perform Multi-party symmetric encryption	Encrypt plaintext data	API return value	API call parameters, key, plaintext	Status, ciphertext	AES for Multi-party Symmetric Encryption/Decryption	User - MPC AES Key: W,E
Perform Multi-party symmetric decryption	Decrypt ciphertext data	API return value	API call parameters, key, ciphertext	Status, plaintext	AES for Multi-party Symmetric Encryption/Decryption	User - MPC AES Key: W,E
Generate Multi-party asymmetric key pair	Generate a public/private key pair	API return value	API call parameters	Status, key pair	ECDSA for Multi-party Asymmetric Key Pair Generation RSA for Multi-party Asymmetric Key Pair Generation	User - MPC ECDSA Public key: G,R - MPC ECDSA Private key: G,R - MPC RSA public key: G,R - MPC RSA private key: G,R
Generate Multi-party digital signature	Generate a digital signature	API return value	API call parameters, key, message	Status, signature	ECDSA for Multi-party Digital Signature Generation RSA for Multi-party Digital Signature Generation	User - MPC ECDSA Private key: W,E - MPC RSA public key: W,E
Perform symmetric encryption	Encrypt plaintext data	API return value	API call parameters, key, plaintext	Status, ciphertext	AES for Symmetric Encryption/Decryption AES-XTS for Symmetric Encryption/Decryption	User - AES key: W,E - AES XTS key: W,E
Perform symmetric decryption	Decrypt ciphertext data	API return value	API call parameters, key, ciphertext	Status, plaintext	AES for Symmetric Encryption/Decryption AES-XTS for Symmetric Encryption/Decryption	User - AES key: W,E - AES XTS key: W,E
Generate symmetric digest	Generate symmetric digest	API return value	API call parameters, key, plaintext	Status, digest	AES-CMAC for Message Authentication AES-GMAC for Message Authentication	User - AES CMAC key: W,E - AES GMAC key: W,E
Verify symmetric digest	Verify symmetric digest	API return value	API call parameters, key, digest	Status	AES-CMAC for Message Authentication AES-GMAC for Message Authentication	User - AES CMAC key: W,E - AES GMAC key: W,E
Perform authenticated symmetric encryption	Encrypt plaintext	API return value	API call parameters, key, plaintext	Status, ciphertext	AES-CCM for Authenticated Symmetric Encryption/Decryption AES-GCM for Authenticated Symmetric Encryption/Decryption	User - AES CCM key: W,E - AES GCM key: W,E - AES GCM IV: G,R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform authenticated symmetric decryption	Decrypt ciphertext	API return value	API call parameters, key, ciphertext	Status, plaintext	AES-CCM for Authenticated Symmetric Encryption/Decryption AES-GCM for Authenticated Symmetric Encryption/Decryption	User - AES CCM key: W,E - AES GCM key: W,E - AES GCM IV: W,E
Generate random number	Generate random bits using DRBG	API return value	API call parameters	Status, random bits	DRBG	User - DRBG entropy input: W,E - DRBG seed: G,E - DRBG 'V' value: G,E - DRBG 'Key' value: G,E
Perform keyed hash operation	Compute a message authentication code	API return value	API call parameters, key, message	Status, MAC	HMAC for Message Authentication	User - HMAC key: W,E
Perform hash operation	Compute a message digest	API return value	API call parameters	Status, hash	SHA for Message Digest SHA3 for Message Digest SHAKE for Extendable Output Function	User
Generate asymmetric key pair	Generate a public/private key pair	API return value	API call parameters	Status, key pair	ECDSA for Key Generation RSA for Key Generation	User - ECDSA public key: G,R - ECDSA private key: G,R - RSA public key: G,R - RSA private key: G,R - ECDH public component: G,R - ECDH private component: G,R
Verify ECDSA public key	Verify an ECDSA public key	API return value	API call parameters, key	Status	ECDSA for Key Verification	User - ECDSA public key: W
Generate digital signature	Generate a digital signature	API return value	API call parameters, key, message	Status, signature	ECDSA for Digital Signature Generation RSA for Signature Generation	User - ECDSA private key: W,E - RSA private key: W,E
Verify digital signature	Verify a digital signature	API return value	API call parameters, key, signature, message	Status	ECDSA for Digital Signature Verification RSA for Signature Verification	User - ECDSA public key: W,E - RSA public key: W,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Perform key wrap	Perform key wrap	API return value	API call parameters, encryption key, key	Status, encrypted key	AES for Key Wrapping/Unwrapping	User - AES key: W,E - AES CCM key: W,E - AES CMAC key: W,E - AES GMAC key: W,E - AES GCM key: W,E - AES GCM IV: G,R,E
Perform key unwrap	Perform key unwrap	API return value	API call parameters, decryption key, key	Status, decrypted key	AES for Key Wrapping/Unwrapping	User - AES key: W,E - AES CCM key: W,E - AES CMAC key: W,E - AES GMAC key: W,E - AES GCM key: W,E - AES GCM IV: W,E
Compute shared secret	Compute ECDH shared secret suitable for use as input to a TLS KDF	API return value	API call parameters	Status, shared secret	ECDH Shared Secret Computation	User - ECDH public component: W,E - ECDH private component: W,E
TLS v1.2 network protocol	Utilize TLS protocol	API return value	API call parameters, TLS certs and keys, raw application data, TLS session data	Status, TLS keys, TLS session data, decrypted application data	TLS v1.2 Key Agreement TLS v1.2 Authentication TLS v1.2 Data Encryption/Decryption	User - ECDH public component: W,E - ECDH private component: W,E - TLS extended pre-master secret: W,E - TLS master secret: W,E - TLS Authentication Key (HMAC key): W,E - TLS Session Key: W,E - TLS Server Authentication Key: R,E
Derive keys via TLS v1.2 KDF	Derive TLS session and integrity keys	API return value	API call parameters, TLS extended pre-master secret	Status, TLS keys	TLS v1.2 Key Derivation	User - TLS extended pre-master secret: W,E - TLS master secret: G,E - TLS Session Key: G,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Certificate Management/Handling	Certificate management services	API return value	API call parameters, AES key, private keys, public keys, certificates, certificate data.	Status	AES for Symmetric Encryption/Decryption ECDSA for Digital Signature Generation ECDSA for Digital Signature Verification RSA for Signature Generation RSA for Signature Verification SHA for Message Digest SHA3 for Message Digest	User - AES key: W,E - AES XTS key: W,E - ECDSA public key: R,W,E - ECDSA private key: R,W,E - RSA public key: R,W,E - RSA private key: R,W,E
Perform key agreement functions	Establish symmetric key using ECDH key agreement	API return value	API call parameters	Status, symmetric key	TLS v1.2 Key Agreement	User - ECDH public component: W,E - ECDH private component: W,E - AES key: G,R - AES GCM key: G,R - AES GCM IV: G,R - HMAC key: G,R
Derive key via HKDF	Derive key from HKDF	API return value	API call parameters, input key material	Status, key	HKDF for Key Derivation	User - HKDF Derived key: G,R
Derive key via PBKDF2	Derive key from PBKDF2	API return value	API call parameters, password	Status, key	PBKDF for Key Derivation	User - Password: W,E - PBKDF Derived key: G,R

Table 11: Approved Services

* Per FIPS 140-3 Implementation Guidance 2.4.C, the **Show Status** and **Show Versioning Information** services do not require a service indicator.

4.4 Non-Approved Services

The module does not offer any non-Approved services.

N/A for this module.

5. Software/Firmware Security

5.1 Integrity Techniques

All software components within the cryptographic boundary are verified using an approved integrity technique implemented within the cryptographic module itself.

The module implements independent HMAC SHA2-256 checks for the integrity test of each library file; failure of the integrity test for any file will cause the module to enter a critical error state.

The calling application is responsible for the initialization process and loading of the library. The module is designed with a default entry point (DEP) that ensures that the pre-operational self-tests are initiated automatically when the module is loaded without action from the module operator.

5.2 Initiate on Demand

The CO can initiate the pre-operational tests on demand by issuing the `securikey_fips_self_test()` call.

6. Operational Environment

6.1 Operational Environment Type and Requirements

The Ideem ZSM Cryptographic Module comprises a software cryptographic library that executes in a Modifiable operational environment.

The cryptographic module has control over its own SSPs. The process and memory management functionality of the host platform'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 environments provide 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 environments. Processes that are spawned by the module are owned by the module and are not owned by external processes/operators.

7. Physical Security

The cryptographic module is a multi-chip standalone software module and does not include physical security mechanisms. Therefore, per section 7.5 of the FIPS PUB 140-3 Management Manual this section is not applicable.

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

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

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

Table 12: Storage Areas

9.2 SSP Input-Output Methods

The table below lists SSP input and output methods for this module.

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
Plaintext import via API parameter	External	RAM	Plaintext	Manual	Electronic	
Plaintext export via API parameter	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.

Zeroization Method	Description	Rationale	Operator Initiation
Zeroize service	The OpenSSL_cleanse() function zeroizes SSPs. OpenSSL_cleanse() may only be called directly by the calling application for some SSPs. However, other SSPs will be zeroized by an indirect call to OpenSSL_cleanse() via object destruction APIs.	The OpenSSL_cleanse() service zeroizes SSPs, which makes them irretrievable.	The operator calls the OpenSSL_cleanse(). The successful completion of the procedural zeroization suffices as the implicit indicator that zeroization has completed.

Table 14: SSP Zeroization Methods

9.4 SSPs

The module supports the keys and other SSPs listed in the table below. Note that all SSP imports and exports are electronic and performed withing the Tested OE's Physical Parameter (TOEPP).

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
MPC AES Key	Multi-party symmetric encryption and decryption	128 bits - 128 bits	Symmetric Key - CSP			AES for Multi-party Symmetric Encryption/Decryption
MPC ECDSA Public key	Public key paired with the MPC ECDSA private key.	256 bits - 128 bits	Public/Private - PSP	ECDSA for Multi-party Asymmetric Key Pair Generation		
MPC ECDSA Private key	Multi-party digital signature generation	256 bits - 128 bits	Public/Private - CSP	ECDSA for Multi-party Asymmetric Key Pair Generation		ECDSA for Multi-party Digital Signature Generation
MPC RSA public key	Public key paired with the MPC ECDSA private key	Between 2048 and 4096 bits - Between 112 and 150 bits	Public/Private - PSP	RSA for Multi-party Asymmetric Key Pair Generation		
MPC RSA private key	Multi-party digital signature generation	Between 2048 and 4096 bits - Between 112 and 150 bits	Public/Private - CSP	RSA for Multi-party Asymmetric Key Pair Generation		RSA for Multi-party Digital Signature Generation
AES key	Symmetric encryption and decryption; key wrap and unwrap Modes: (CBC, CFB, CTR, ECB, OFB, KW, KWP)	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP		TLS v1.2 Key Derivation AES for Key Wrapping/Unwrapping	AES for Symmetric Encryption/Decryption
AES CCM key	Authenticated symmetric encryption, decryption; key transport	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP		TLS v1.2 Key Derivation AES for Key Wrapping/Unwrapping	AES-CCM for Authenticated Symmetric Encryption/Decryption
AES GCM key	Authenticated symmetric encryption, decryption; key transport	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP		TLS v1.2 Key Derivation	AES-GCM for Authenticated Symmetric Encryption/Decryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES GCM IV	Initialization vector for AES GCM	96 bits - n/a	Initialization Vector - CSP	Generated internally in compliance with the provisions of a peer-to-peer industry standard protocol. Technique 1.a. as specified in FIPS 140-3 IG C.H.		AES-GCM for Authenticated Symmetric Encryption/Decryption
AES XTS key	Symmetric encryption, decryption	256-bits - 256-bits	Symmetric Key - CSP			AES-XTS for Symmetric Encryption/Decryption
AES CMAC key	Key wrap and unwrap	Between 128 and 256 bits - Between 128 and 256 bits	Symmetric Key - CSP			AES-CMAC for Message Authentication
AES GMAC key	MAC generation, verification	Between 128 and 256 bits - Between 128 and 256 bits	Authentication Key - CSP			AES-GMAC for Message Authentication
ECDH private component	ECDH shared secret computation	Between 224 and 521 bits - Between 112 and 256 bits	Public/Private - CSP	ECDSA for Key Generation		ECDH Shared Secret Computation TLS v1.2 Key Agreement
ECDH public component	ECDH shared secret computation	Between 224 and 521 bits - Between 112 and 256 bits	Public/Private - PSP	ECDSA for Key Generation		ECDH Shared Secret Computation TLS v1.2 Key Agreement
DRBG entropy input	Entropy material for DRBG	Between 128 and 512 bits - Between 128 and 512 bits	Entropy Input - CSP			DRBG
DRBG seed	Seeding material for DRBG	Between 256 and 384 - Between 256 and 384	Seed - CSP	DRBG		DRBG
DRBG 'V' value	State values for DRBG	128 bits - 128 bits	DRBG state - CSP	DRBG		DRBG

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
DRBG 'Key' value	State values for DRBG	Between 128 and 256 bits - Between 128 and 256 bits	DRBG state - CSP	DRBG		DRBG
ECDSA private key	Digital signature generation	Between 224 and 521 bits - Between 112 and 256 bits	Public/Private - CSP	ECDSA for Key Generation		ECDSA for Digital Signature Generation
ECDSA public key	Digital signature generation	Between 224 and 521 bits - Between 112 and 256 bits	Public/Private - PSP	ECDSA for Key Generation		ECDSA for Digital Signature Verification
HMAC key	Keyed hash	224 bits (minimum) - 112 bits (minimum)	Authentication - CSP		TLS v1.2 Key Derivation	HMAC for Message Authentication AES for Key Wrapping/Unwrapping
RSA private key	Digital signature generation	Between 2048 and 4096 bits - Between 112 and 150 bits	Public/Private - CSP	RSA for Key Generation		RSA for Signature Generation
RSA public key	Digital signature verification	Between 2048 and 4096 bits - Between 112 and 150 bits	Public/Private - PSP	RSA for Key Generation		RSA for Signature Verification
TLS extended pre-master secret	Derivation of the TLS master secret	384 bits - 384 bits	Pre-master secret - CSP		TLS v1.2 Key Agreement	TLS v1.2 Key Derivation
TLS master secret	Derivation of the AES key, AES-GCM key, and HMAC key used for securing TLS connections	384 bits - 384 bits	Master secret - CSP		TLS v1.2 Key Agreement	TLS v1.2 Key Derivation
TLS Session Key	Encryption and decryption of TLS session packets	128 or 256 bits - 128 or 256 bits	Symmetric key - CSP		TLS v1.2 Key Derivation	TLS v1.2 Data Encryption/Decryption
TLS Authentication Key (HMAC key)	Authentication of TLS session packets	Between 160 and 384 bits - Between 160 and 384 bits	Authentication - CSP		TLS v1.2 Key Derivation	TLS v1.2 Data Encryption/Decryption

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
TLS Server Authentication Key	Digital signature verification	2048 bits - 128 bits	Public - PSP			TLS v1.2 Authentication
HKDF Derived key	Symmetric encryption and decryption	256 bits - 256 bits	Symmetric key - CSP		HKDF for Key Derivation	
PBKDF Derived key	Symmetric encryption and decryption; Storage application only	256 bits - 256 bits	Symmetric key - CSP		PBKDF for Key Derivation	
Password	Input to PBKDF for key derivation	n/a - n/a	Password - CSP			PBKDF for Key Derivation

Table 15: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
MPC AES Key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	
MPC ECDSA Public key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	MPC ECDSA Private key:Paired With
MPC ECDSA Private key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	MPC ECDSA Public key:Paired With
MPC RSA public key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	MPC RSA private key:Paired With
MPC RSA private key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	MPC RSA public key:Paired With
AES key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	
AES CCM key	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	
AES GCM key	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	AES GCM IV:Other
AES GCM IV	Plaintext export via API parameter	RAM:Plaintext		Zeroize service	
AES XTS key	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	
AES CMAC key	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	
AES GMAC key	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
ECDH private component	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	ECDH public component:Paired With
ECDH public component	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	ECDH private component:Paired With
DRBG entropy input	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	
DRBG seed		RAM:Plaintext		Zeroize service	
DRBG 'V' value		RAM:Plaintext		Zeroize service	
DRBG 'Key' value		RAM:Plaintext		Zeroize service	
ECDSA private key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	ECDSA public key:Paired With
ECDSA public key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	ECDSA private key:Paired With
HMAC key	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	
RSA private key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	RSA public key:Paired With
RSA public key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	RSA private key:Paired With
TLS extended pre-master secret	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	
TLS master secret		RAM:Plaintext		Zeroize service	
TLS Session Key		RAM:Plaintext		Zeroize service	
TLS Authentication Key (HMAC key)		RAM:Plaintext		Zeroize service	
TLS Server Authentication Key	Plaintext import via API parameter Plaintext export via API parameter	RAM:Plaintext		Zeroize service	
HKDF Derived key	Plaintext export via API parameter	RAM:Plaintext		Zeroize service	
PBKDF Derived key	Plaintext export via API parameter	RAM:Plaintext		Zeroize service	
Password	Plaintext import via API parameter	RAM:Plaintext		Zeroize service	

Table 16: SSP Table 2

10. Self-Tests

The module performs pre-operational self-tests and conditional self-tests. Pre-operational tests are performed between the time the cryptographic module is instantiated and before the module transitions to the operational state. Conditional self-tests are performed by the module during module operation when certain conditions exist. The following sections list the self-tests performed by the module, their expected error status, and the error resolutions.

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

10.1 Pre-Operational Self-Tests

The module performs the following pre-operational self-test(s):

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
HMAC-SHA2-256 (A5056)	SHA2-256	Software Integrity	SW/FW Integrity	A boolean value is returned indicating the success (true) or failure (false) of the self-test procedure call.	Software integrity test for libcrypto
HMAC-SHA2-256 (A5056)	SHA2-256	Software Integrity	SW/FW Integrity	A boolean value is returned indicating the success (true) or failure (false) of the self-test procedure call.	Software integrity test for libssl
HMAC-SHA2-256 (A5056)	SHA2-256	Software Integrity	SW/FW Integrity	A boolean value is returned indicating the success (true) or failure (false) of the self-test procedure call.	Software integrity test for libsecurikey

Table 17: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

The module performs the following conditional self-tests:

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB (A5055)	128-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Encrypt	Triggered upon first usage of MPC services.
ECDSA SigGen (FIPS186-4) (A5055)	P-256; SHA2-256	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Sign	Triggered upon first usage of MPC services.
RSA SigGen (FIPS186-5) (A5055)	2048-bit; SHA2-224; PKCS#1.5 scheme	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Sign	Triggered upon first usage of MPC services.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB (A5056)	128-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Encrypt	After successful completion of software integrity tests.
AES-ECB (A5056)	128-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Decrypt	After successful completion of software integrity tests.
AES-CCM (A5056)	192-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Encrypt	After successful completion of software integrity tests.
AES-CCM (A5056)	192-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Decrypt	After successful completion of software integrity tests.
AES-GCM (A5056)	128-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Encrypt	After successful completion of software integrity tests.
AES-GCM (A5056)	128-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Decrypt	After successful completion of software integrity tests.
AES-XTS Testing Revision 2.0 (A5056)	128-bit, 256-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Encrypt	After successful completion of software integrity tests.
AES-XTS Testing Revision 2.0 (A5056)	128-bit, 256-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Decrypt	After successful completion of software integrity tests.
AES-CMAC (A5056)	CBC mode; 128-bit, 192-bit, 256-bit	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Generate	After successful completion of software integrity tests.
Counter DRBG (A5056)	AES, 256-bit, with derivation function	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Instantiate, Reseed, Generate	After successful completion of software integrity tests.
ECDSA SigGen (FIPS186-5) (A5056)	P-224; SHA2-256	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Sign	After successful completion of software integrity tests.
ECDSA SigVer (FIPS186-5) (A5056)	P-224; SHA2-256	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Verify	After successful completion of software integrity tests.
RSA SigGen (FIPS186-5) (A5056)	2048-bit; SHA2-256; PKCS#1.5 scheme	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Sign	After successful completion of software integrity tests.
RSA SigVer (FIPS186-5) (A5056)	2048-bit; SHA2-256; PKCS#1.5 scheme	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Verify	After successful completion of software integrity tests.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
HMAC-SHA-1 (A5056)	SHA-1	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hashed message authentication	Upon power-up and before the pre-operational software integrity tests.
HMAC-SHA2-224 (A5056)	SHA2-224	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hashed message authentication	Upon power-up and before the pre-operational software integrity tests.
HMAC-SHA2-256 (A5056)	SHA2-256	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hashed message authentication	Upon power-up and before the pre-operational software integrity tests.
HMAC-SHA2-384 (A5056)	SHA2-384	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hashed message authentication	Upon power-up and before the pre-operational software integrity tests.
HMAC-SHA2-512 (A5056)	SHA2-512	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hashed message authentication	Upon power-up and before the pre-operational software integrity tests.
SHA-1 (A5056)	-	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hash	Upon power-up and before the pre-operational software integrity tests.
SHA2-224 (A5056)	-	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hash	Upon power-up and before the pre-operational software integrity tests.
SHA2-256 (A5056)	-	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hash	Upon power-up and before the pre-operational software integrity tests.
SHA2-384 (A5056)	-	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hash	Upon power-up and before the pre-operational software integrity tests.
SHA2-512 (A5056)	-	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hash	Upon power-up and before the pre-operational software integrity tests.
SHA3-256 (A5056)	-	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Hash	Upon power-up and before the pre-operational software integrity tests.
KAS-ECC-SSC Sp800-56Ar3 (A5056)	P-224	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Shared Secret "Z" Computation	After successful completion of software integrity tests.
KDA HKDF SP800-56Cr2 (A5056)	SHA2-256	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Derive	After successful completion of software integrity tests.
PBKDF (A5056)	SHA2-256	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Derive	After successful completion of software integrity tests.

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
TLS v1.2 KDF RFC7627 (A5056)	SHA2-256	KAT	CAST	A boolean value indicating success (true) or failure (false) of the self-test procedure call.	Derive	After successful completion of software integrity tests.
ECDSA KeyGen (FIPS186-4) (A5055)	-	PCT	PCT	An integer value indicating success (1) or failure (0) of the self-test procedure call.	Sign/Verify	Executed upon key pair generation before returning key pair.
RSA KeyGen (FIPS186-5) (A5055)	-	PCT	PCT	An integer value indicating success (1) or failure (0) of the self-test procedure call.	Sign/Verify	Executed upon key pair generation before returning key pair.
ECDSA KeyGen (FIPS186-5) (A5056)	-	PCT	PCT	An integer value indicating success (1) or failure (0) of the self-test procedure call.	Sign/Verify	Executed upon key pair generation before returning key pair.
RSA KeyGen (FIPS186-5) (A5056)	-	PCT	PCT	An integer value indicating success (1) or failure (0) of the self-test procedure call.	Sign/Verify	Executed upon key pair generation before returning key pair.
ECDH	-	PCT	PCT	An integer value indicating success (1) or failure (0) of the self-test procedure call.	Sign/Verify	Executed upon key pair generation before returning key pair.
AES-XTS Testing Revision 2.0 (A5056)	-	Duplicate Key Test	Critical Function	An integer value indicating success (1) or failure (0) of the self-test procedure call.	Duplicate key test	Executed upon initialization of AES-XTS cipher with key data.

Table 18: Conditional Self-Tests

10.3 Periodic Self-Test Information

The module does not implement automatic periodic self-tests, however the operator can perform all module self-tests on demand by issuing the `securikey_fips_self_test()` call.

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
HMAC-SHA2-256 (A5056)	Software Integrity	SW/FW Integrity	On Demand	Manually
HMAC-SHA2-256 (A5056)	Software Integrity	SW/FW Integrity	On Demand	Manually
HMAC-SHA2-256 (A5056)	Software Integrity	SW/FW Integrity	On Demand	Manually

Table 19: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB (A5055)	KAT	CAST	On Demand	Manually
ECDSA SigGen (FIPS186-4) (A5055)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-5) (A5055)	KAT	CAST	On Demand	Manually
AES-ECB (A5056)	KAT	CAST	On Demand	Manually
AES-ECB (A5056)	KAT	CAST	On Demand	Manually

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CCM (A5056)	KAT	CAST	On Demand	Manually
AES-CCM (A5056)	KAT	CAST	On Demand	Manually
AES-GCM (A5056)	KAT	CAST	On Demand	Manually
AES-GCM (A5056)	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A5056)	KAT	CAST	On Demand	Manually
AES-XTS Testing Revision 2.0 (A5056)	KAT	CAST	On Demand	Manually
AES-CMAC (A5056)	KAT	CAST	On Demand	Manually
Counter DRBG (A5056)	KAT	CAST	On Demand	Manually
ECDSA SigGen (FIPS186-5) (A5056)	KAT	CAST	On Demand	Manually
ECDSA SigVer (FIPS186-5) (A5056)	KAT	CAST	On Demand	Manually
RSA SigGen (FIPS186-5) (A5056)	KAT	CAST	On Demand	Manually
RSA SigVer (FIPS186-5) (A5056)	KAT	CAST	On Demand	Manually
HMAC-SHA-1 (A5056)	KAT	CAST	On Demand	Manually
HMAC-SHA2-224 (A5056)	KAT	CAST	On Demand	Manually
HMAC-SHA2-256 (A5056)	KAT	CAST	On Demand	Manually
HMAC-SHA2-384 (A5056)	KAT	CAST	On Demand	Manually
HMAC-SHA2-512 (A5056)	KAT	CAST	On Demand	Manually
SHA-1 (A5056)	KAT	CAST	On Demand	Manually
SHA2-224 (A5056)	KAT	CAST	On Demand	Manually
SHA2-256 (A5056)	KAT	CAST	On Demand	Manually
SHA2-384 (A5056)	KAT	CAST	On Demand	Manually
SHA2-512 (A5056)	KAT	CAST	On Demand	Manually
SHA3-256 (A5056)	KAT	CAST	On Demand	Manually
KAS-ECC-SSC Sp800-56Ar3 (A5056)	KAT	CAST	On Demand	Manually
KDA HKDF SP800-56Cr2 (A5056)	KAT	CAST	On Demand	Manually
PBKDF (A5056)	KAT	CAST	On Demand	Manually
TLS v1.2 KDF RFC7627 (A5056)	KAT	CAST	On Demand	Manually
ECDSA KeyGen (FIPS186-4) (A5055)	PCT	PCT	n/a	n/a
RSA KeyGen (FIPS186-5) (A5055)	PCT	PCT	n/a	n/a
ECDSA KeyGen (FIPS186-5) (A5056)	PCT	PCT	n/a	n/a
RSA KeyGen (FIPS186-5) (A5056)	PCT	PCT	n/a	n/a
ECDH	PCT	PCT	n/a	n/a
AES-XTS Testing Revision 2.0 (A5056)	Duplicate Key Test	Critical Function	n/a	n/a

Table 20: Conditional Periodic Information

10.4 Error States

The tables below describe the error states the status indicators of the module.

Name	Description	Conditions	Recovery Method	Indicator
Critical Error	The module immediately terminates the calling application's API call. Subsequent requests made by the calling application for cryptographic services will return failure indicator, disabling all access to cryptographic functions, SSPs, and data output services.	If the module fails pre-operational integrity tests, the SHA/HMAC pre-operational KATs (SHA, HMAC), or the conditional CASTs (DRBG, AES-ECB, AES-CCM, AES-GCM, AES-XTS, AES-CMAC, ECDSA Sign/Verify, RSA Sign/Verify, KAS-ECC Shared Secret "Z" Computation, HKDF, PBKDF, TLS v1.2 KDF).	The module must be re-instantiated by the calling application. If errors persist, the CO should contact Ideem, Inc. for assistance.	Returned error code and sets an internal flag. Further requests will return this failure indicator.
Soft Error	The module enters this state upon the failure of a PCT or self-test. The module transitions back to normal operation where the service requiring the self-test can be re-run or a new service can be performed.	If the module fails ECDSA/RSA/ECDH PCTs or the AES-XTS duplicate key test.	Module records the error and resumes normal operation	Returns error code

Table 21: Error States

11. Life-Cycle Assurance

The sections below describe how to ensure the module is operating in its validated configuration, including the following:

- Procedures for secure installation, initialization, startup, and operation of the module
- Maintenance requirements
- Administrator and non-Administrator guidance

Operating the module without following the guidance herein (including the use of undocumented services) will result in non-compliant behavior and is outside the scope of this Security Policy.

11.1 Installation, Initialization, and Startup Procedures

11.1.1 Secure Installation

As the module is an integrated component of Ideem's product application software, module operators have no ability to independently load the module onto the target platform. The module is distributed to the end operator as part of an SDK[2] developed by Ideem. The module is distributed as a package containing the pre-compiled binaries and HMAC digest files. The module and its calling application are to be installed on a platform specified in section 2.2 or one where portability is maintained. For correct operation the module must be installed on both a server RHEL environment) and client (iOS or Android) side. Ideem does not provide any mechanisms to directly access the module, its source code, its APIs, or any information sent between it and other Ideem applications.

11.1.2 Initialization

This module is designed to support the Ideem application solely, and this application are the sole consumers of the cryptographic services provided by the module. No end-user action is required to initialize the module for operation; the calling application performs any actions required to initialize the module.

The pre-operational integrity test and conditional CASTs are performed automatically via a default entry point (DEP) when the module is loaded for execution, without any specific action from the calling application or the end-user. End-users have no means to short-circuit or bypass these actions. Failure of any of the initialization actions will result in a failure of the module to load for execution.

11.1.3 Startup

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

[2] SDK – Software Development Kit

11.2 Administrator Guidance

There are no specific management activities required of the CO role to ensure that the module runs securely. If any irregular activity is observed, or if the module is consistently reporting errors, then Ideem Customer Support should be contacted.

The following list provides additional guidance for the CO:

- The CO can initiate the pre-operational self-tests and conditional CASTs on demand for periodic testing of the module by re-instantiating the module.

- The CO may call the `securikey_fips_self_test()` API command to initiate the pre-operational self-tests and conditional CASTs on demand for MPC algorithms.

- The `fips_post_status()` API command returns TRUE(1) if the module's self-tests have passed but returns FALSE(0) otherwise.

- The `securikey_fips_get_module_information()` API command returns the module's versioning information as a string.

11.3 Non-Administrator Guidance

The following list provides additional policies for the User role:

- The cryptographic module's services are designed to be provided to a calling application. Excluding the use of the NIST-defined elliptic curves as trusted third-party domain parameters, all other assurances from *FIPS PUB 186-4* (including those required of the intended signatory and the signature verifier) are outside the scope of the module and are the responsibility of the calling application.

- 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 length of a single data unit encrypted or decrypted with the AES-XTS shall not exceed 2^{20} AES blocks; that is, 16 MB of data per AES-XTS instance. An XTS instance is defined in section 4 of NIST SP 800-38E. The AES-XTS mode shall only be used for the cryptographic protection of data on storage devices. The AES-XTS shall not be used for other purposes, such as the encryption of data in transit. The module implements the check to ensure that the two AES keys used in the XTS-AES algorithm are not identical.

- The module supports importing GCM IVs generated externally for decryption purposes. The operator shall not perform AES GCM encryption when the IV is provided from outside the cryptographic boundary of the module. Importing an external AES GCM IV will result in a non-conformance.

12. Mitigation of Other Attacks

12.1 Attack List

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

12.2 Mitigation Effectiveness

Secure multi-party computation is used to generate and split a cryptographic key into two or more key shares, such that knowledge of a key share provides no information about the logical key. The original key is never stored nor recomputed, so leakage of the key share from a single module gives an attacker no knowledge about the private keying material used by that module.

12.3 Guidance and Constraints

There is no further guidance or constraints.

12.4 Additional Information

There is no additional information.

Appendix A. Acronyms and Abbreviations

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

Table 22. Acronyms and Abbreviations

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CBC	Cipher Block Chaining
CCCS	Canadian Centre for Cyber Security
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
DES	Data Encryption Standard
DH	Diffie-Hellman
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EMI/EMC	Electromagnetic Interference /Electromagnetic Compatibility
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode
GMAC	Galois Message Authentication Code
GPC	General-Purpose Computer
HMAC	(keyed-) Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
KTS	Key Transport Scheme
KW	Key Wrap
KWP	Key Wrap with Padding
NIST	National Institute of Standards and Technology

Acronym	Definition
OS	Operating System
PCT	Pairwise Consistency Test
PKCS	Public Key Cryptography Standard
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	Special Publication
TDES	Triple Data Encryption Standard

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