FIPS 140-2 Security Policy
SafeZone FIPS Cryptographic Module

Rambus Global Inc., Finnish branch
Sokerilinnantie 11 C
FI-02600 Espoo
Finland
Phone: +358 50 3560966

Rambus Inc.
1050 Enterprise Way
Sunnyvale
CA 94089
United States

2020-03-13

Revision C

Software Version 1.1.0

Document Number: FIPS-2020-1022

Modification History

2020-03-13	Policy revision C: FIPS Lib 1.1.0 policy, added more vendor affirmed platforms, updated vendor contact information
2015-05-27	Policy revision B: FIPS Lib 1.1.0 policy
2014-12-12	Policy revision A: FIPS Lib 1.1.0 policy, based on FIPS Lib 1.0.3 (A)
2014-05-12	Policy revision D: Revalidation
2014-05-07	Updated according to NIST SP 800-131A
2014-05-02	Added more vendor affirmed platforms
2014-04-25	Added several vendor affirmed platforms
2014-04-25	Added validated one platform: Samsung Galaxy Note 3 (ARMv7-a)
2013-12-31	Updated contact addresses
2013-03-15	Policy revision C: The original validation

FIPS 140-2 Security Policy
SafeZone FIPS Cryptographic Module

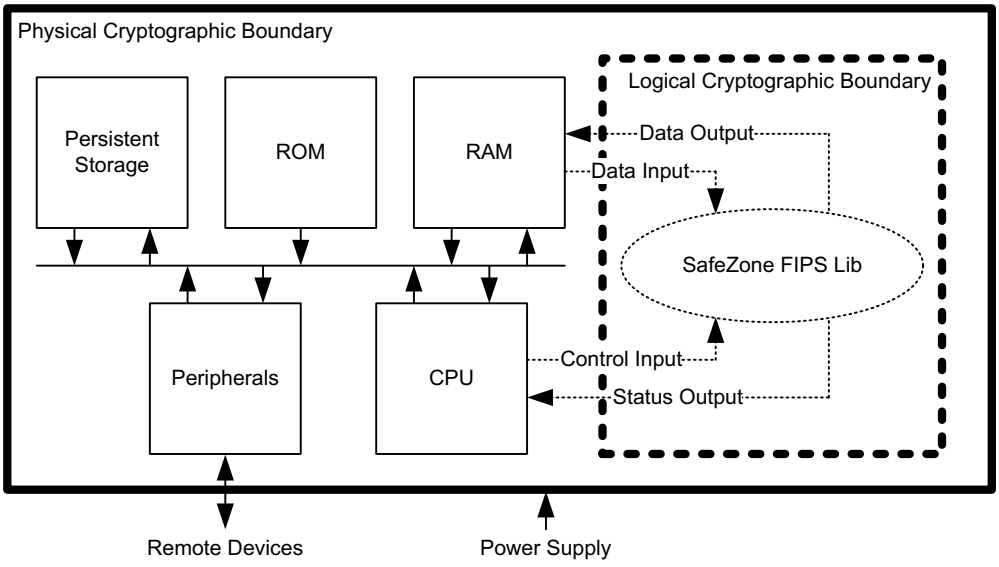
1 Introduction

SafeZone FIPS Cryptographic Module is a FIPS 140-2 Security Level 1 validated software cryptographic module from Rambus. This module is a toolkit that provides the most commonly used cryptographic primitives for a wide range of applications, including primitives needed for VPN (Virtual Private Network), TLS (Transport Layer Security), DAR (Data-At-Rest), and DRM (Digital Rights Management) clients.

SafeZone FIPS Cryptographic Module is a software-based product with a custom, small-footprint API (Application Programming Interface). The cryptographic module has been designed to provide the necessary cryptographic capabilities for other Rambus products. However, it can also be used stand-alone in custom-developed products to provide the required cryptographic functionality.

The module is primarily intended for embedded products with a general-purpose operating system.

Figure 1: SafeZone FIPS Cryptographic Module Cryptographic Boundary



For FIPS 140-2 purposes, SafeZone FIPS Cryptographic Module is classified as a multi-chip standalone cryptographic module. Within the *logical* boundary of SafeZone FIPS Cryptographic Module is the `libsafezone-sw-fips.a/so` object code library, also known as SafeZone FIPS Lib. The *physical* cryptographic boundary

of the module is the enclosure of a general-purpose computing device executing the application that embeds the SafeZone FIPS Cryptographic Module.

The SafeZone FIPS Cryptographic Module has been tested for validation on the following platforms:

Processor / Tested Platform	Operating System	Version
ARMv6 / Raspberry Pi	Linux[1] / kernel 3.10 (single-user mode)	1.1.0
ARMv7-a / Arndale	<t-base 300 (single-user mode)	1.1.0
ARMv7-a / Samsung Galaxy Note 3	Android 4.4 (single-user mode)	1.1.0
Intel Atom Z2560 / Samsung Galaxy Tab 3 10.1	Android 4.2 (single-user mode)	1.1.0
Apple A7 (64-bit) / iPad Mini with Retina Display	iOS 7.1 (single-user mode)	1.1.0
Apple A7 (32-bit) / iPad Mini with Retina Display	iOS 7.1 (single-user mode)	1.1.0
Intel Atom Z3740 with AES-NI / ASUS Transformer	Linux / kernel 3.13 (single-user mode)	1.1.0
Intel Atom Z3740 (64-bit) / ASUS Transformer	Linux / kernel 3.13 (single-user mode)	1.1.0
Intel Atom Z3740 (64-bit) with AES-NI / ASUS Transformer	Linux / kernel 3.13 (single-user mode)	1.1.0

Compliance is maintained on platforms for which the binary executable remains unchanged. The module has been confirmed by the vendor to be operational on the following platforms. As allowed by the FIPS 140-2 Implementation Guidance G.5, the validation status of the Cryptographic Module is maintained when operated in the following additional operating environments:

Implementation Guidance G.5 Recompilation	
Processor / Device	**Operating System**
ARMv7-a / Nexus 5	Android 5.0-5.1 (single-user mode)
ARMv7-a / Nexus 6	Android 5.0-5.1 (single-user mode)
NVidia Tegra K1 (ARMv8-a; 64-bit) / Nexus 9	Android 5.0 (single-user mode)
NVidia Tegra K1 (32-bit) / Nexus 9	Android 5.0 (single-user mode)
ARMv7-a / Fuji Xerox 000T789485	Wind River Linux 6.0 (single-user mode)
ARMv7-a (32-bit)	Android 6.0
ARMv8-a (64-bit)	Android 7.0-7.1

[1]Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. All other brands and product names are trademarks or registered trademarks of their respective owners.

ARMv8-a (64-bit)	Android 8.0-8.1
ARMv8-a (64-bit)	Android 9.0
ARMv8-a (64-bit)	Android 10.0
ARMv8-a (64-bit)	Yocto Linux 2.7
ARMv8-a (64-bit)	Linux / kernel 4.9
ARMv8-a (64-bit)	Linux / kernel 4.14
ARM926EJ-S	Nucleus 3.0
Intel Xeon X86-32	PhotonOS 2.0
Intel Xeon X86-32	Ubuntu Linux 18.04
Intel Xeon X86-64	Ubuntu Linux 18.04
Intel Xeon X86-64	Yocto Linux 2.6
Intel Core X86-32	Ubuntu Linux 18.04
Intel Core X86-64	Ubuntu Linux 18.04
Intel Core X86-64	Yocto Linux 2.6
ARMv7-a (32-bit) / Raspberry Pi 3	Rasbian Linux / kernel 4.19

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

1.1 Purpose

The purpose of this document is to describe the secure operation of the SafeZone FIPS Cryptographic Module including the initialization, roles, and responsibilities of operating the product in a secure, in FIPS 140 mode of operation.

1.2 Security level

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS 140-2.

Security Level	
Security Requirements Specification	**Level**
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

1.3 Glossary

Term/Acronym	Description
AES	Advanced Encryption Standard
API	Application Programming Interface
CMVP	Cryptographic Module Validation Program (FIPS 140)
CSP	Critical Security Parameter
DEP	Default Entry Point
DRM	Digital Rights Management
DSS	Digital Signature Standard
EC	Elliptic Curve
FIPS	Federal Information Processing Standard
IKE	Internet Key Exchange
KEM	Key-Encapsulation Mechanism (See NIST SP 800-56B)
KTS	Key Transport Scheme
OAEP	Optimal Asymmetric Encryption Padding
PRF	Pseudo-Random Function
SHS	Secure Hash Standard
SRDI	Security Relevant Data Item
TLS	Transport Layer Security
Triple-DES	Triple Data Encryption Standard
VPN	Virtual Private Network

2 Ports and Interfaces

As a software-only module, the SafeZone FIPS Cryptographic Module provides an API logical interface for invocation of FIPS140-2 approved cryptographic functions. The functions shall be called by the referencing application, which assumes the operator role during application execution. The API, through the use of input parameters, output parameters, and function return values, defines the four FIPS 140-2 logical interfaces: data input, data output, control input and status output.

Logical Interfaces	API
Data Input	The data read from memory area(s) provided to the invoked function via parameters that point to the memory area(s).
Control Input	The API function invoked and function parameters designated as control inputs.
Data Output	The data written to memory area(s) provided to the invoked function via parameters that point to the memory area(s).
Status Output	The return value of the invoked API function.
Power Interface	Not accessible via the API. The power interface is used as applicable on the physical device.

3 Roles, Services, and Authentication

The SafeZone FIPS Cryptographic Module supports the *Crypto Officer* and *User* roles. The operator of the module will assume one of these two roles. Only one role may be active at a time. The Crypto Officer role is assumed implicitly upon module installation, uninstallation, initialization, zeroization, and power-up self-testing. If initialization and self-testing are successful, a transition to the User role is allowed and the User will be able to use all keys and cryptographic operations provided by the module, and to create any CSPs (except Trusted Root Key CSPs which may only be created in the Crypto Officer role).

The four unique run-time services given only to the Crypto Officer role are the ability to initialize the module, to set-up key material for Trusted Root Key CSP(s), to modify the entropy source, and to switch to the User role to perform any activities allowed for the User role. The SafeZone FIPS Cryptographic Module does not support concurrent operators.

3.1 Roles and Services

The module does not authenticate the operator role.

3.1.1 User Role

The User role is assumed once the Crypto Officer role is finished with module initialization and explicitly switches the role using the `FL_LibEnterUserRole` API function. The User role is intended for common cryptographic use. The full list of cryptographic services available to the User role is supplied in chapter 5 of this document.

Service	Description
All services except installation, initialization, entropy source nomination, and creation of Trusted Root Key CSPs.	All standard cryptographic operations of the module, such as symmetric encryption, message authentication codes, and digital signatures. The User role may also allocate the key assets and load values for any of these cryptographic purposes. The SafeZone FIPS Cryptographic Module also provides a 'Show Status' service (API function `FL_LibStatus`) that can be used to query the current status of the cryptographic module. A macro based on `FL_LibStatus` is provided (`FL_IS_IN_APPROVED_MODE`), which returns true if the module is currently in an approved mode of operation.

3.1.2 Crypto-officer Role

The Crypto Officer role can perform all the services allowed for the User role plus a handful of additional ones. Separate from the run-time services of the module, the tasks of installing and uninstalling the module to and from the host system imply the role of a Crypto Officer. The four run-time services available only to the Crypto Officer are initializing the module for use, creating key material for Trusted Root Key CSPs, modifying the entropy source, and switching to the User role.

Service	Description
All services allowed for User role	See above.
Initialization	Loading and preparing the module for use.
Trusted Root Key creation	Load key material into the module for local security purposes (`FL_RootKeyAllocateAndLoadValue`).

Entropy Source	Select the provider of the external entropy source. (`FL_RbgInstallEntropySource`, `FL_RbgRequestSecurityStrength`, `FL_RbgUseNonblockingEntropySource`).
Switch to the User Role	Uses the `FL_LibEnterUserRole` API function to switch to User role.
Installation	When the module is installed to a host system.
Uninstallation	When the module is removed from a host system.

3.2 Authentication Mechanisms and Strength

FIPS 140-2 Security Level 1 does not require *role-based* or *identity-based* operator authentication. The SafeZone FIPS Cryptographic Module will not authenticate the operator.

4 Secure Operation and Security Rules

In order to operate the SafeZone FIPS Cryptographic Module securely, the operator should be aware of the security rules enforced by the module and should adhere to the rules for physical security and secure operation.

4.1 Security Rules

To operate the SafeZone FIPS Cryptographic Module securely, the operator of the module must follow these instructions:

1. The operating environment that executes the SafeZone FIPS Cryptographic Module must ensure single operator mode of operation to be compliant with the requirements for the FIPS 140-2 Level 1.
2. The correct operation of the module depends on the Default Entry Point. It is not allowed to prevent execution of the Default Entry Point (the function `FL_LibInit`).
3. The operator must not call `ptrace` or `strace` functions, or run `gdb` or other debugger when the module is in the FIPS mode.
4. If the hardware platform has a connector for an external debugger (for example JTAG), that connector must not be used while the module is in FIPS mode.
5. The SafeZone FIPS Cryptographic Module keeps all CSPs and other protected objects in Random Access Memory (RAM). The operator(s) must only use these objects via the handles provided by the SafeZone FIPS Cryptographic Module. It is not permissible to directly access these objects in the memory.
6. The operator must not call functions provided by the SafeZone FIPS Cryptographic Module that are not explicitly specified in the appropriate guidance document for User or Crypto Officer.
7. When using cryptographic services provided by the SafeZone FIPS Cryptographic Module, the operator must follow the appropriate guidance for each cryptographic algorithm. Although the cryptographic algorithms provided by the SafeZone FIPS Cryptographic Module are recommended or allowed by NIST, secure operation of these algorithms requires thorough understanding of the recommendations and appropriate limitations.
8. The SafeZone FIPS Cryptographic Module aims to be flexible and therefore it includes support for cryptographic algorithms or key lengths that were considered secure until 2013 according to NIST SP 800-131A. It is the responsibility of the SafeZone FIPS Cryptographic Module user to ensure that disallowed algorithms or key lengths are not used.
9. Some of the implemented cryptographic algorithms offer key lengths exceeding the current NIST specifications. Such key lengths must not be used, unless following newer guidance from NIST.
    a. RSA Key Pair Generation provided by the module (FIPS 186-3 B.3.6) is only FIPS-approved for RSA modulus sizes of 2048 bits and 3072 bits. It is not permissible to generate keys using other RSA modulus sizes.
10. The Crypto Officer must ensure that the Trusted Root Key has sufficient entropy to meet all FIPS 140-2 requirements for its usage in the module.

4.2 Physical Security Rules

The physical device on which the SafeZone FIPS Cryptographic Module is executed must follow the physical security rules applicable to the purpose of the device. The SafeZone FIPS Cryptographic Module is software-based and does not provide physical security.

4.3 Secure Operation Initialization Rules

The SafeZone FIPS Cryptographic Module must be linked with an application to become executable. The software code of the module (the `libsafezone-sw-fips.a` object code library or the `libsafezone-sw-fips.so` dynamically loadable library) is linked with an end application producing an executable application for the target platform. The application is installed in a platform-specific way, e.g. when purchased from an application store for the platform. In some cases there is no need for installation, e.g. when a mobile equipment vendor includes the application with the equipment.

The SafeZone FIPS Cryptographic Module is loaded by loading an application that links the library statically. The SafeZone FIPS Cryptographic Module is initialized automatically upon loading. On some platforms the module is implemented as a dynamically loadable module. In this case, the module is loaded as needed by the dynamic linker.

The SafeZone FIPS Cryptographic Module does not support operator authentication and thus does not require any authentication itself. The SafeZone FIPS Cryptographic Module is by default in FIPS-approved mode once initialized. Usually, the module does not require any special set-up or initialization except for installation.

5 Definition of SRDIs (Security Relevant Data Items) Modes of Access

This chapter specifies security relevant data items as well as the access control policy that is enforced by the SafeZone FIPS Cryptographic Module.

Each SRDI is held in the asset store accompanied by a security usage policy. The policy is set when the asset is allocated with `FL_RootKeyAllocateAndLoadValue`, `FL_AssetAllocate`, `FL_AssetAllocateBasic`, `FL_AssetAllocateSamePolicy` or `FL_AssetAllocateAndAssociateKeyExtra`. When the asset is accessed for use in a cryptographic operation, the policy is tested to ensure that the asset is eligible for the requested use. A policy typically consists of the allowed algorithm(s), the allowed strength of the algorithm, and the direction of the operation (encryption or decryption).

5.1 FIPS Approved and Allowed algorithms

The SafeZone FIPS Cryptographic Module implements the following FIPS-approved algorithms:

Algorithm	Implementation Details	Algorithm Certificate(s)
RSA FIPS 186-4 Signature Generation Key Pair Generation	2048, and 3072 bit keys; PKCS #1 v1.5 and PSS; SHA-224, SHA-256, SHA-384, SHA-512	RSA #1593
RSA FIPS 186-4 Signature Validation	1024, 2048, and 3072 bit keys; PKCS #1 v1.5 and PSS	RSA #1593
DSA FIPS 186-4 Signature Generation Domain Parameter Generation Key Pair Generation	P=2048/N=224, P=2048/N=256, P=3072/N=256; SHA-224, SHA-256, SHA-384, SHA-512	DSA #905
DSA FIPS 186-4 Signature Validation Domain Parameter Validation	P=1024/N=160, P=2048/N=224, P=2048/N=256, P=3072/N=256	DSA #905
ECDSA FIPS 186-4 Signature Generation Key Pair Generation	NIST P-224, P-256, P-384 and P-521 curves; SHA-224, SHA-256, SHA-384, SHA-512	ECDSA #567

Algorithm	Implementation Details	Algorithm Certificate(s)
ECDSA FIPS 186-4 Signature Validation Public Key Verification	NIST P-192, P-224, P-256, P-384 and P-521 curves	ECDSA #567
AES FIPS 197, NIST SP 800-38A	128, 192, 256 bit keys; ECB, CBC, CTR mode	AES #3123
AES CCM NIST SP 800-38C	128, 192, 256 bit keys	AES #3123
AES GCM NIST SP 800-38D	128, 192, 256 bit keys	AES #3123
XTS-AES NIST SP 800-38E	256, 512 bit keys (128-bit or 256-bit encryption strength)	AES #3123
Triple-DES NIST SP 800-67	192 bit keys; ECB and CBC mode	Triple-DES #1793
CMAC NIST SP 800-38B	128, 192, 256 bit keys	AES #3123
HMAC FIPS 198-1	112-512 bit keys; SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	HMAC #1980
SHS FIPS 180-3	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512; BYTE only	SHS #2599
DRBG NIST SP 800-90	AES-128-CTR without df or reseed AES-256-CTR with df and reseed	DRBG #634, DRBG #637
KTS (KEM NIST SP 800-56B)	2048, 3072 bit keys; RSA-KEM-KWS-basic (section 9.3.3); vendor affirmed; key-wrapping; key establishment methodology provides 112 bits or 128 bits of encryption strength	N/A, Vendor-affirmed
KTS (OAEP NIST SP 800-56B)	2048, 3072 bit keys; RSA-OAEP (section 9.2.3); vendor affirmed; key-wrapping; key establishment methodology provides 112 bits or 128 bits of encryption strength	N/A, Vendor-affirmed
PBKDF NIST SP 800-132	with SHA-1, SHA-256	N/A, Vendor-affirmed

Algorithm	Implementation Details	Algorithm Certificate(s)
KDF NIST SP 800-108	112-512 bit keys; SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, AES-CMAC; counter, feedback and double pipeline modes	KBKDF #37, KBKDF #38, KBKDF #39, KBKDF #40 Key derivation methodology provides between 112 and 256 bits of encryption strength.
Application Specific Key Derivation Functions NIST SP 800-135rev1	IKEv1 Key Derivation Functions IKEv2 Key Derivation Functions TLS 1.0/1.1 Key Derivation Functions TLS 1.2 Key Derivation Functions	CVL #385
FFC Diffie-Hellman primitive; A part of NIST SP 800-56A	Key Agreement Primitives; 2048, 3072 bit modular Diffie-Hellman groups	CVL #384 Key establishment methodology provides 112 bits or 128 bits of encryption strength.
ECC CDH primitive; A part of NIST SP 800-56A	Key Agreement Primitives; NIST P-224, P-256, P-384 and P-521 curves	CVL #384 Key establishment methodology provides between 112 and 256 bits of encryption strength.

Algorithm	Implementation Details	Algorithm Certificate(s)
KTS (NIST SP 800-38F Key Wrapping)	Key Wrapping function KW CIPH=AES; 128, 192, 256 bit keys Key Wrapping function KWP CIPH=AES; 128, 192, 256 bit keys	KTS (AES Cert. #3123) Key establishment methodology provides between 128 and 256 bits of encryption strength

The cryptographic module supports the following non-approved algorithms in the approved mode of operation as allowed:

Algorithm	Algorithm Type	Utilization
RSA Encryption (PKCS #1 v1.5)	Key Transport; 2048, 3072 bit keys	(RSA Cert. #1593) Key establishment methodology provides 112 bits or 128 bits of encryption strength.
MD5	Message Digest; This function is only allowed as a part of an approved key transport scheme (e.g. TLS 1.0 or TLS 1.1).	
/dev/random	Non-Approved RBG	An entropy source for NIST SP 800-90 DRBG.
/dev/urandom	Non-Approved RBG	An entropy source for NIST SP 800-90 DRBG.

<t-base-300 RNG	Non-Approved RBG	An entropy source for NIST SP 800-90 DRBG.

The SafeZone FIPS Cryptographic Module is intended for products where FIPS 140-2 approved algorithms are used. Rambus also provides solutions for customers that need software or hardware based implementations for non-approved cryptographic algorithms, such as Camellia and C2. However, to ensure that SafeZone FIPS Cryptographic Module remains the most convenient solution for products required to be FIPS 140-2 approved, it does not implement these algorithms.

5.2 Non-FIPS mode of operation

In the end of 2013, some of algorithms previously allowed by the NIST were disallowed. This was because 80-bits of security was considered no longer sufficient. See document NIST SP 800-131A for details. The SafeZone FIPS Cryptographic Module implements additional key lengths for some of these algorithms (RSA, DSA, ECDSA) for compatibility with applications previously using these key sizes. These no longer approved key sizes shall only be used in non-FIPS mode of operation.

The non-FIPS validated algorithms and key sizes supported by the module are:

Algorithm	Implementation Details	Reason for algorithm being no longer allowed in FIPS mode.
RSA FIPS 186-2 Signature Generation	1024, 1536, 2048, 3072, and 4096 bit keys; PKCS #1 v1.5 and PSS	Transition from FIPS 186-2 to 186-4.
RSA FIPS 186-4 Signature Generation Key Pair Generation	1024 bit keys; PKCS #1 v1.5 and PSS	Key length used provides less than 112 bits of encryption strength
DSA FIPS 186-4 Signature Generation Domain Parameter Generation Key Pair Generation	P=1024/N=160	Key length used provides less than 112 bits of encryption strength
ECDSA FIPS 186-2/4 Signature Generation Key Pair Generation	NIST P-192 curve	Key length used provides less than 112 bits of encryption strength

Algorithm	Implementation Details	Reason for algorithm being no longer allowed in FIPS mode.
ECDSA FIPS 186-2 Signature Generation	NIST P-224, P-256, P-384 and P-521 curves	Transition from FIPS 186-2 to 186-4.
HMAC FIPS 198-1	80-104 bit keys; SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	Key length used provides less than 112 bits of encryption strength.
KTS (KEM NIST SP 800-56B)	1024, 1536, bit keys; RSA-KEM-KWS-basic; key-wrapping	Key establishment methodology provides less than 112 bits of encryption strength
KTS (OAEP NIST SP 800-56B)	1024, 1536 bit keys; RSA-OAEP; key-wrapping	Key establishment methodology provides less than 112 bits of encryption strength
KDF NIST SP 800-108	80-104 bit keys; SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, AES-CMAC; counter, feedback and double pipeline modes	Key derivation methodology provides less than 112 bits of encryption strength.
FFC Diffie-Hellman primitive;	Key Agreement Primitives; 1024 bit modular Diffie-Hellman groups	Key establishment methodology provides less than 112 bits of encryption strength.
ECC CDH primitive; A part of NIST SP 800-56A	Key Agreement Primitives; NIST P-192 curves	Key establishment methodology provides less than 112 bits of encryption strength.
RSA Encryption (PKCS #1 v1.5)	Key Transport; 1024, 1536 bit keys	Key establishment methodology provides less than 112 bits of encryption strength.

5.3 Cryptographic Keys, CSPs, and SRDIs

While operating in a FIPS-compliant manner, the asset store within the SafeZone FIPS Cryptographic Module may contain the following security relevant data items (depending on which keys will be used by the user):

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
General Keys/CSPs						
AES Encryption Key	AES including modes ECB, CBC, and CTR	128, 192, 256 bits	Key created for the purposes of encrypting and/or decrypting data using AES algorithm	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
AES CCM Encryption Key	AES CCM	128, 192, 256 bits	Key created for the purposes of authenticated encryption and/or decryption of data using AES and CCM algorithms	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
AES GCM Encryption Key	AES GCM	128, 192, 256 bits	Key created for the purposes of authenticated encryption and/or decryption of data using AES and GCM algorithms	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
XTS-AES Encryption Key	XTS-AES	256, 512 bits	Key created for the purposes of encrypting and/or decrypting data using AES algorithm in XTS mode	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
Triple-DES Encryption Key	Triple-DES	192 bits	Key created for the purposes of encrypting and/or decrypting data using Triple-DES algorithm	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
CMAC Key	CMAC + AES	128, 192, 256 bits	Key created for the purposes of generating and verifying CMAC authentication codes	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
CMAC Verify Key	CMAC + AES	128, 192, 256 bits	Key created for the purpose of verifying CMAC authentication codes	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
KDF Key Derivation key	NIST SP 800-108 + HMAC or CMAC	112-512 bits	Key created for the purpose of deriving other keys as specified in NIST SP 800-108 or IKEv1/IKEv2 key derivation specified in NIST SP 800-135.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
TLS-PRF Key Derivation Key	NIST SP 800-135	112-512 bits	Key created for the purpose of key derivation using TLS1.0/TLS1.2 key derivation function presented in NIST SP 800-135.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
HMAC Key	HMAC + SHS	112-512 bits	Key created for the purposes of generating and verifying HMAC authentication codes	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
HMAC Verify Key	HMAC + SHS	112-512 bits	Key created for the purpose of verifying HMAC authentication codes	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
RSA Signing Key	RSA Private Key (CRT)	2048, 3072 bits (modulus size)	Private key for the purpose of signing data using RSA with PKCS #1v1.5 or PSS padding.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
DSA Signing Key	DSA Private Key	P=2048/N=224, P=2048/N=256, P=3072/N=256	Private key for the purpose of signing data using DSA algorithm. Includes associated domain parameters.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
ECDSA Signing Key	ECDSA Private Key	P-224, P-256, P-384, P-521	Private key for the purpose of signing data using ECDSA algorithm	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
AES Key-Wrapping Key	AES	128, 192, 256 bits	Key created for the purposes of data or key wrapping and unwrapping using NIST SP 800-38F algorithm	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
Diffie-Hellman Private Value	Diffie-Hellman	P=2048/N=224, P=2048/N=256, P=3072/N=256	Private value for the purpose of key agreement using Diffie-Hellman algorithm. Includes associated domain parameters.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
EC Diffie-Hellman Private Value	EC Diffie-Hellman	P-224, P-256, P-384, P-521	Private value for the purpose of key agreement using Elliptic Curve Diffie-Hellman algorithm.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
KTS (KEM) Unwrapping Key	RSA Private Key (CRT)	2048, 3072 bits	Private key for the purpose of transporting keys using RSA with KEM as specified in NIST SP 800-56B	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
KTS (OAEP) Unwrapping Key	RSA Private Key (CRT)	2048, 3072 bits	Private key for the purpose of transporting keys using RSA with OAEP as specified in NIST SP 800-56B	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
KTS (PKCS #1 v1.5) RSA Unwrapping Key	RSA Private Key (CRT)	2048, 3072 bits	Private key for the purpose of transporting keys using RSA with PKCS #1 v1.5 padding (also known as RSA Encryption)	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
Trusted Keys						
Trusted Root Key	NIST SP 800-108 KDF	256 bits	Key used for deriving other keys as per NIST SP 800-108. Can only derive 'Trusted KDK' and 'Trusted KEKDK' keys.	Crypto Officer	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
Trusted KDK	NIST SP 800-108 KDF	256 bits	Key used for deriving other keys as per NIST SP 800-108.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
Trusted KEKDK	NIST SP 800-108 KDF + AES (Key Wrap)	256 bits	Key used for wrapping keys with combination of NIST SP800-108 KDF and AES Key Wrap.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
Other CSPs						
DRBG CTR-128 state: Key	CTR_DRBG 128-bits with derivation function	128 bits	Key for DRBG used for random number and key/key pair generation purposes.	Entropy source	Plaintext in RAM	Power Off, FL_LibUnInit

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
DRBG CTR-128 state: V	CTR_DRBG 128-bits with derivation function	128 bits	V value for DRBG used for random number and key/key pair generation purposes.	Entropy source	Plaintext in RAM	Power Off, FL_LibUnInit
DRBG CTR-256 state: Key	CTR_DRBG 256-bits with derivation function	256 bits	Key for DRBG used for random number and key/key pair generation purposes.	Entropy source	Plaintext in RAM	Power Off, FL_LibUnInit
DRBG CTR-256 state: V	CTR_DRBG 256-bits with derivation function	128 bits	V value for DRBG used for random number and key/key pair generation purposes.	Entropy source	Plaintext in RAM	Power Off, FL_LibUnInit
Public Keys						
Software Integrity Public Key	ECDSA / Verify	NIST P-224	Public key used by Power-on Software Integrity to ensure the integrity of the Cryptographic Module.	Embedded in the software	Plaintext in persistent storage	none
RSA Verification Key	RSA Public Key	1024, 2048, 3072 bits modulus size	Public key for the purpose of verifying signed data using RSA with PKCS #1 v1.5 or PSS padding. Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
DSA Verification Key	DSA Public Key	P=1024/N=160, P=2048/N=224, P=2048/N=256, P=3072/N=256	Public key for the purpose of verifying signed data using DSA algorithm. Includes associated domain parameters. Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
ECDSA Verification Key	ECDSA Public Key	P-192, P-224, P-256, P-384, P-521	Public key for the purpose of verifying signed data using ECDSA algorithm. Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
Diffie-Hellman Public Value	Diffie-Hellman	P=2048/N=224, P=2048/N=256, P=3072/N=256	Public value for the purpose of key agreement using the Diffie-Hellman algorithm. Includes associated domain parameters. Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
EC Diffie-Hellman Public Value	EC Diffie-Hellman	P-224, P-256, P-384, P-521	Public value for the purpose of key agreement using the Elliptic Curve Diffie-Hellman algorithm. Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
KTS (KEM) Wrapping Key	RSA Public Key	2048, 3072 bits	Public key for the purpose of transporting keys using RSA with KEM as specified in NIST SP 800-56B. Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
KTS (OAEP) Wrapping Key	RSA Public Key	2048, 3072 bits	Public key for the purpose of transporting keys using RSA with OAEP as specified in NIST SP 800-56B. Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
KTS (PKCS #1 v1.5) RSA Wrapping Key	RSA Public Key	2048, 3072 bits	Public key for the purpose of transporting keys using RSA with PKCS #1 v1.5 padding (also known as RSA Encryption). Not considered sensitive or CSP.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit

All the cryptographic keys and other security relevant materials handled by the module can be zeroized by using the cryptographic module, with the exception of the Software Integrity Public Key that is used in the self-test to validate the module.

There are three ways to zeroize a key: individual keys can be explicitly zeroized using the `FL_AssetFree` function call, all keys are zeroized once the module is uninitialized (`FL_LibUnInit`) or encounters error state, and (as all the keys handled by the module except the Software Integrity Public key are stored in RAM memory), the keys can also be zeroized by turning the power off.

The main difference between normal and Trusted Keys is that Trusted Keys do not allow the User role to pick the key material to use, but the keys can only be derived from the trusted root key provided by the Crypto Officer role. The primary use of trusted keys is wrapping and unwrapping other keys for purposes of persistent storage outside the SafeZone FIPS Cryptographic Module. Trusted Keys do not provide any additional security for FIPS purposes. They merely are identifiers for the keys derived from the trusted root key.

5.4 Access Control Policy

The module allows controlled access to the SRDIs contained within it. The following table defines the access that an operator or an application has to each SRDI while operating the SafeZone FIPS Cryptographic Module in a given role performing a specific service (command). The permissions are categorized as a set of four separate permissions: read [R] (the SRDI can be read by this operation), write [W] (the SRDI can be written by this operation), execute [X] (the SRDI can be used in this operation), and delete [D] (the SRDI will be zeroized by this operation). If no permission is listed, then an operator outside the SafeZone FIPS Cryptographic Module has no access to the SRDI.

The operations are presented in the following tables: for secret keys, private keys, public keys, and none (operations which do not affect any of SRDI). The operations which are not appropriate for a specific key type have been omitted.

SafeZone FIPS Cryptographic Module SRDI/Role/Service Access Policy **Secret Keys**	Security Relevant Data Item	AES Encryption Key	AES CCM Encryption Key	AES GCM Encryption Key	XTS-AES Encryption Key	Triple-DES Encryption Key	CMAC Key	CMAC Verify Key	KDF Key Derivation key	TLS-PRF Key Derivation key	HMAC Key	HMAC Verify Key	AES Key-Wrapping Key	Trusted Root Key	Trusted KDK	Trusted KEKDK	DRBG state: Key / V
Role/Service																	
User role or Crypto Officer Role																	
Zeroize (FL_LibUnInit)		D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
Create Key (FL_AssetAllocate, FL_AssetAllocateBasic, FL_AssetAllocateSamePolicy, FL_AssetAllocateAndAssociateKeyExtra, FL_AssetLoadValue, FL_AssetLoadMultipart, FL_AssetLoadMultipartConvertBigInt)		W	W	W	W	W	W	W	W	W	W	W	W				
Copy Key (FL_AssetCopy)		W	W	W	W	W	W	W	W	W	W	W	W				
Delete Key (FL_AssetFree)		D	D	D	D	D	D	D	D	D	D	D	D		D	D	
Examine Key (FL_AssetShow, FL_AssetCheck)																	
Generate Key (FL_AssetLoadRandom)		W	W	W	W	W	W	W	W	W	W	W	W				XW
Bulk Encryption/Decryption (FL_CipherInit, FL_CipherContinue, FL_CipherFinish)		X			X	X											

SafeZone FIPS Cryptographic Module SRDI/Role/Service Access Policy **Secret Keys**	Security Relevant Data Item	AES Encryption Key	AES CCM Encryption Key	AES GCM Encryption Key	XTS-AES Encryption Key	Triple-DES Encryption Key	CMAC Key	CMAC Verify Key	KDF Key Derivation key	TLS-PRF Key Derivation key	HMAC Key	HMAC Verify Key	AES Key-Wrapping Key	Trusted Root Key	Trusted KDK	Trusted KEKDK	DRBG state: Key / V
Role/Service																	
Authenticated Encryption/Decryption with Associated Data (FL_EncryptAuthInitRandom, FL_EncryptAuthInitDeterministic, FL_CryptAuthInit[2], FL_CryptGcmAadContinue, FL_CryptGcmAadFinish, FL_CryptAuthContinue, FL_EncryptAuthFinish, FL_EncryptAuthPacketFinish, FL_DecryptAuthFinish)			X	X													
MAC Generation (FL_MacGenerateInit, FL_MacGenerateContinue, FL_MacGenerateFinish)							X				X						
MAC Verification (FL_MacVerifyInit, FL_MacGenerateContinue, FL_MacGenerateFinish)							X	X			X	X					
DRBG Random Number Generation (FL_RbgGenerateRandom)																	XW
DRBG Reseeding (FL_RbgReseed)																	XW
Key Derivation (FL_KeyDeriveKdk)		W	W	W	W	W	W	W	XW	W	W	W	W				
TLS-PRF Key Derivation (FL_KeyDeriveKdk, FL_DeriveTlsPrf)		W	W	W	W	W	W	W	W	XW	W	W	W				
IKEv1 Key Derivation (FL_IkePrfExtract, FL_IKEv1ExtractSKEYID_DSA, FL_IKEv1ExtractSKEYID_PSK, FL_IKEv1ExtractSKEYID_PKE, FL_IKEv1DeriveKeyingMaterial)		W	W	W	W	W	W	W	XW	W	W	W	W				
IKEv2 Key Derivation (FL_IkePrfExtract, FL_IKEv2ExtractSKEYSEED, FL_IKEv2ExtractSKEYSEEDrekey, FL_IKEv2DeriveDKM)		W	W	W	W	W	W	W	XW	W	W	W	W				
AES Key Wrapping (FL_AssetsWrapAes, FL_AssetsWrapAes38F)		R	R	R	R	R	R	R	R	R	R	R	XR				
AES Key Unwrapping (FL_AssetsUnwrapAes, FL_AssetsUnwrapAes38F)		W	W	W	W	W	W	W	W	W	W	W	XW				
AES Data Wrapping (FL_CryptKw)													X				
AES Data Unwrapping (FL_CryptKw)													X				
Trusted Root Key Derivation (FL_TrustedKdkDerive, FL_TrustedKekdkDerive)														X	W	W	
Trusted KDK Key Derivation (FL_TrustedKeyDerive)		W	W	W	W	W	W	W	W	W	W	W	W		X		
Trusted Key Wrapping (FL_AssetWrapTrusted)		R	R	R	R	R	R	R	R	R	R	R	R			X	

[2] Function may only be used to begin AES-CCM encryption operation or to continue multipacket operation with deterministic IV. In particular, the function shall not be used to initialize AES-GCM encryption.

SafeZone FIPS Cryptographic Module SRDI/Role/Service Access Policy **Secret Keys**	Security Relevant Data Item	AES Encryption Key	AES CCM Encryption Key	AES GCM Encryption Key	XTS-AES Encryption Key	Triple-DES Encryption Key	CMAC Key	CMAC Verify Key	KDF Key Derivation key	TLS-PRF Key Derivation key	HMAC Key	HMAC Verify Key	AES Key-Wrapping Key	Trusted Root Key	Trusted KDK	Trusted KEKDK	DRBG state: Key / V
Role/Service																	
Trusted Key Unwrapping (FL_AssetUnwrapTrusted)		W	W	W	W	W	W	W	W	W	W	W	W			X	
PBKDF2 Key Derivation (FL_KeyDerivePbkdf2)		W	W	W	W	W	W	W	W	W	W	W	W				
Crypto-officer Role																	
Entropy Source Installation (FL_RbgInstallEntropySource, FL_RbgRequestSecurityStrength, FL_RbgUseNonblockingEntropySource)																	W
Create Trusted Root Key (FL_RootKeyAllocateAndLoadValue)														W			

SafeZone FIPS Cryptographic Module SRDI/Role/Service Access Policy **Private Keys**	Security Relevant Data Item	RSA Signing Key	DSA Signing Key	ECDSA Signing Key	Diffie-Hellman Private Value	EC Diffie-Hellman Private Value	KTS (KEM) Unwrapping Key	KTS (OAEP) Unwrapping Key	KTS (PKCS #1 v1.5) RSA Unwrapping Key	DRBG state: Key / V
Role/Service										
User role or Crypto Officer Role										
Zeroize (FL_LibUnInit)		D	D	D	D	D	D	D	D	D
Create Key (FL_AssetAllocate, FL_AssetAllocateBasic, FL_AssetAllocateSamePolicy, FL_AssetAllocateAndAssociateKeyExtra, FL_AssetLoadValue, FL_AssetLoadMultipart, FL_AssetLoadMultipartConvertBigInt)		W	W	W	W	W	W	W	W	
Copy Key (FL_AssetCopyValue)		W	W	W	W	W	W	W	W	
Delete Key (FL_AssetFree)		D	D	D	D	D	D	D	D	
Examine Key (FL_AssetShow, FL_AssetCheck)										
Generate Key (FL_AssetLoadRandom)										X W
Generate Key Pair (FL_AssetGenerateKeyPair)		W	W	W	W	W	W	W	W	X W
DSA/Diffie-Hellman Domain Parameter and Key Pair Generation (FL_AssetGenerateKeyPair)			W		W					X W

SafeZone FIPS Cryptographic Module

SRDI/Role/Service Access Policy

Private Keys

Role/Service	Security Relevant Data Item	RSA Signing Key	DSA Signing Key	ECDSA Signing Key	Diffie-Hellman Private Value	EC Diffie-Hellman Private Value	KTS (KEM) Unwrapping Key	KTS (OAEP) Unwrapping Key	KTS (PKCS #1 v1.5) RSA Unwrapping Key	DRBG state: Key / V
Signature Generation (FL_HashSignFips186, FL_HashSignPkcs1, FL_HashSignPkcs1Pss)		X	X	X						X W
AES Key Wrapping (FL_AssetsWrapAes, FL_AssetsWrapAes38F)		R	R	R	R	R	R	R	R	
AES Key Unwrapping (FL_AssetsUnwrapAes, FL_AssetsUnwrapAes38F)		W	W	W	W	W	W	W	W	
Trusted Key Wrapping (FL_AssetWrapTrusted)		R	R	R	R	R	R	R	R	
Trusted Key Unwrapping (FL_AssetUnwrapTrusted)		W	W	W	W	W	W	W	W	
PBKDF2 Key Derivation (FL_KeyDerivePbkdf2)		W	W	W	W	W	W	W	W	
Diffie-Hellman Key Agreement (FL_DeriveDh)					X					
Elliptic Curve Diffie-Hellman Key Agreement (FL_DeriveDh)						X				

SafeZone FIPS Cryptographic Module

SRDI/Role/Service Access Policy

Public Keys

Role/Service	Security Relevant Data Item	Software Integrity Public Key	RSA Verification Key	DSA Verification Key	ECDSA Verification Key	Diffie-Hellman Public Value	EC Diffie-Hellman Public Value	KTS (KEM) Wrapping Key	KTS (OAEP) Wrapping Key	KTS (PKCS #1 v1.5) RSA Wrapping Key	DRBG state: Key / V
User role or Crypto-Officer Role											
Zeroize (FL_LibUnInit)			D	D	D	D	D	D	D	D	D
On-demand self-test (FL_LibSelfTest)		X									
Create Key (FL_AssetAllocate, FL_AssetAllocateBasic, FL_AssetAllocateSamePolicy, FL_AssetAllocateAndAssociateKeyExtra, FL_AssetLoadValue, FL_AssetLoadMultipart, FL_AssetLoadMultipartConvertBigInt)			W	W	W	W	W	W	W	W	
Copy Key (FL_AssetCopyValue)			W	W	W	W	W	W	W	W	
Delete Key (FL_AssetFree)			D	D	D	D	D	D	D	D	
Examine Key (FL_AssetShow, FL_AssetCheck)			RX	RX	RX	RX	RX	RX	RX	RX	
Generate Key Pair (FL_AssetGenerateKeyPair)			W	W	W	W	W	W	W	W	X W

SafeZone FIPS Cryptographic Module SRDI/Role/Service Access Policy **Public Keys**	Security Relevant Data Item	Software Integrity Public Key	RSA Verification Key	DSA Verification Key	ECDSA Verification Key	Diffie-Hellman Public Value	EC Diffie-Hellman Public Value	KTS (KEM) Wrapping Key	KTS (OAEP) Wrapping Key	KTS (PKCS #1 v1.5) RSA Wrapping Key	DRBG state: Key / V
Role/Service											
DSA/Diffie-Hellman Domain Parameter and Key Pair Generation (FL_AssetGenerateKeyPair)				W		W					X W
Public Key Validation (FL_AssetCheck)			X	X	X	X	X	X	X	X	
DSA/Diffie-Hellman Domain Parameter Verification (FL_AssetCheck)				X		X					
Signature Verification (FL_HashVerifyFips186, FL_HashVerifyPkcs1, FL_HashVerifyRecoverPkcs1, FL_HashVerifyPkcs1Pss)			X	X	X						
Diffie-Hellman Key Agreement (FL_DeriveDh)						X					
Elliptic Curve Diffie-Hellman Key Agreement (FL_DeriveDh)							X				
Crypto-officer Role											
Module Initialization (FL_LibInit) (This function is automatically invoked upon loading the module)		X									X W

SafeZone FIPS Cryptographic Module SRDI/Role/Service Access Policy **Services not using any SRDI**
Role/Service
User role or Crypto Officer Role
Show Status (FL_LibStatus)
Digest Generation (FL_HashInit, FL_HashContinue, FL_HashFinish, FL_HashFinishKeep, FL_HashSingle)

SafeZone FIPS Cryptographic Module

SRDI/Role/Service Access Policy

Non-FIPS 140-2 Security Relevant Services

Role/Service
Services provided for convenience which offer no FIPS 140-2 Security Relevant Functions
Show Version (FL_LibVersion)
Test DRBG (FL_RbgTestVector)
Check Free Space in Key Store (FL_AssetStoreStatus)

5.5 User Guide

Some of the FIPS Publications or NIST Special Publications require that the Cryptographic Module Security Policy mentions important configuration items for those algorithms. The user of the module shall observe these rules.

5.5.1 NIST SP 800-108: Key Derivation Functions

All three key derivation functions, Counter Mode, Feedback Mode and Double-Pipeline Iteration Mode are supported.

5.5.2 NIST SP 800-132: Password-Based Key Derivation Function

The key derived using NIST SP 800-132 shall only be used for storage purposes.

Both options presented in NIST SP 800-132 for deriving the Data Protection Key from the Master Key are supported.

The SafeZone FIPS Lib does not limit the length of the passphrase used in NIST SP 800-132 PBKDF key derivation. The upper bound for the strength of passwords usually used is between 5 or 6 bits per character. Thus, for security over 64 bits, the passwords must generally be longer than 12 characters.

5.5.3 NIST SP 800-38D: Galois/Counter Mode

The FIPS 140-2 Implementation Guidance A.5 applies to AES-GCM usage with this module.

Item 1 in IG A.5 forbids using external IV for encryption via the `FL_CryptAuthInit` function. However, the `FL_CryptAuthInit` function is still used for decryption and the `FL_CryptAuthInit` function is used for subsequent encryption operations for operation sequences started with the `FL_EncryptAuthInitDeterministic` function.

The operator must use the `FL_EncryptAuthInitRandom` function if random IV generation (IG A.5 item 2) is required, or in case of deterministic IV generation (IG A.5 item 3), the `FL_EncryptAuthInitDeterministic` function.

Note: If IV is generated internally in a deterministic manner, then FIPS 140-2 Implementation Guidance A.5: Item B3 applies: In case a module's power is lost and then restored, the key used for the AES GCM encryption/decryption must be re-distributed.

5.5.4 NIST SP 800-90: Deterministic Random Bit Generator

The module generates cryptographic keys whose strengths are modified by available entropy. No assurance of the minimum strength of the generated keys is given by the module. Depending on the platform, the module provides access to different entropy sources.

By default, the SafeZone FIPS Cryptographic Module DRBG uses `/dev/random` as the entropy source on platforms that provide such an entropy device. This entropy generation path is merely a convenience default. The quality of entropy coming from `/dev/random` is not measured by the SafeZone FIPS Cryptographic Module.

It is possible to use function `FL_RbgUseNonblockingEntropySource` to configure `/dev/urandom` as the entropy source. When using `/dev/urandom` as the entropy source, the module assumes the quality of the entropy source to be 128 bits. The difference between `/dev/random` and `/dev/urandom` is that when the entropy source does not know if there is sufficient entropy available, `/dev/random` will block and `/dev/urandom` will generate pseudo-random values based on available entropy. The quality of entropy coming from `/dev/urandom` is not measured by the SafeZone FIPS Cryptographic Module.

If Crypto Officer uses `/dev/random` or `/dev/urandom` as entropy source, it is up to Crypto Officer to configure it suitably to provide reasonable security. Crypto Officer can provide an entropy function which overrides the default entropy source.

5.5.4.1 iOS entropy source

iOS operating system (e.g. Apple iPad and iPhone devices) `/dev/random` and `/dev/urandom` entropy sources use the Yarrow random number generator. Yarrow includes SHA-1 (160 bits) as a part of its operation and thus entropy generated by the RNG can be considered to be at most 160 bits.

The module generates cryptographic keys whose strengths are modified by available entropy. The encryption strength of the generated cryptographic keys is at most 160 bits.

5.5.4.2 <t-base-300 OS

On <t-base-300 operating environment, devices `/dev/random` and `/dev/urandom` are not available. The <t-base-300 API provides function

`tlApiRandomGenerateData`, which generates cryptographically secure random numbers when generator type `TLAPI_ALG_SECURE_RANDOM` is requested. The cryptographic module uses this function to obtain entropy. The quality of entropy coming from the function is not measured by the SafeZone FIPS Cryptographic Module.

6 Self Tests

6.1 Power-Up Self-Tests

The SafeZone FIPS Cryptographic Module includes the following power-up self tests:

- Software Integrity Test (using ECDSA Verify with NIST P-224)
- KAT test for SHA-1
- KAT test for SHA-512
- KAT test for HMAC SHA-256
- KAT test for AES encryption (CBC, 128-bit key)
- KAT test for AES decryption (CBC, 128-bit key)
- KAT test for AES encryption (CCM, 128-bit key)
- KAT test for AES decryption (CCM, 128-bit key)
- KAT test for AES encryption (GCM, 128-bit key)
- KAT test for AES decryption (GCM, 128-bit key)
- KAT test for AES encryption (XTS, 128-bit key strength)
- KAT test for AES decryption (XTS, 128-bit key strength)
- KAT test for CMAC, 192-bit key
- KAT test for Triple-DES encryption (CBC, 192-bit key)
- KAT test for Triple-DES decryption (CBC, 192-bit key)
- KAT for RSA 2048-bit (PKCS #1 v1.5)
- KAT for DSA (signing P=2048/N=256; verification P=1024/N=160)
- KAT for ECDSA Signing (NIST P-224)
- KAT for KTS: RSA Key Wrapping 2048-bit (RSA-OAEP)
- KAT for Diffie-Hellman
- KAT for EC Diffie-Hellman
- AES-CTR-256 DRBG self-test

The self-tests are invoked automatically upon loading the SafeZone FIPS Cryptographic Module. The initialization function `FL_LibInit` is executed via DEP (default entry point) as specified in FIPS 140-2 Implementation Guidance 9.10.

Any error during the power-up self tests will result in a module transition to the error state. There are two possible ways to recover from the error state:

- Reinitializing the module with the API function sequences `FL_LibUnInit` and `FL_LibInit`.

- Power-cycling the device and reinitialize the module with the API function `FL_LibInit`.

The `FL_LibStatus` API function can be used to obtain the module status. It returns `FL_STATUS_INIT` when the module has not yet been initialized and `FL_STATUS_ERROR` when the module is in error state.

As it is recommended to self-test cryptographic components (like DRBG) frequently, the module provides the capability to invoke the self-tests manually (on demand) with the `FL_LibSelfTest` API function. The important difference between the manually invoked self-tests and the automatically invoked self-tests when initializing the module is that the manually invoked self-tests will not cause zeroization of the key material currently loaded in the module, providing the tests execute successfully.

In general, if a self-test fails, the module will transition to the error state and the return value (status) of the invoked API function will be something other than `FLR_OK`, depending on the current situation.

6.2 Conditional Self tests

The SafeZone FIPS Cryptographic Module contains the following conditional self-tests:

- Pair-wise consistency check for key pairs created for digital signature purposes (DSA, FIPS 186-3)
- Pair-wise consistency check for key pairs created for digital signature purposes (RSA, FIPS 186-3)
- Pair-wise consistency check for key pairs created for digital signature purposes (ECDSA, FIPS 186-3)
- Continuous random number generator test.
- Continuous random number generator test for non-Approved RBG `/dev/random`.
- Continuous random number generator test for non-Approved RBG `/dev/urandom`.

The conditional self-tests for manual key entry and software/firmware load or bypass are not provided, as these are not applicable.

Any error during the conditional self tests will result in a module transition to the error state. The ways to recover from the error state are listed in section 6.1.

7 Mitigation of Other Attacks

The module contains an implementation of the RSA algorithm with data independent processing time for signing and decryption operations. This makes it harder to attack the RSA implementation via timing attacks.

The module does not mitigate other attacks outside the scope of FIPS 140-2.