FIPS 140-2 Security Policy
SafeZone FIPS Cryptographic Module

INSIDE Secure B.V.
(formerly a division of AuthenTec Inc.)
Boxtelseweg 26A
5261 NE Vught
The Netherlands
Phone: +31-73-6581900
Fax: +31-73-6581999

INSIDE Secure
Corporate Headquarters
41 Parc Club du Golf
13856 Aix-en-Provence, France
Phone: +33 (0)4 42 39 63 00

2013-03-15

Revision C

Software Version 1.0.3

Document Number: 001-921100-407

FIPS 140-2 Security Policy
SafeZone FIPS Cryptographic Module

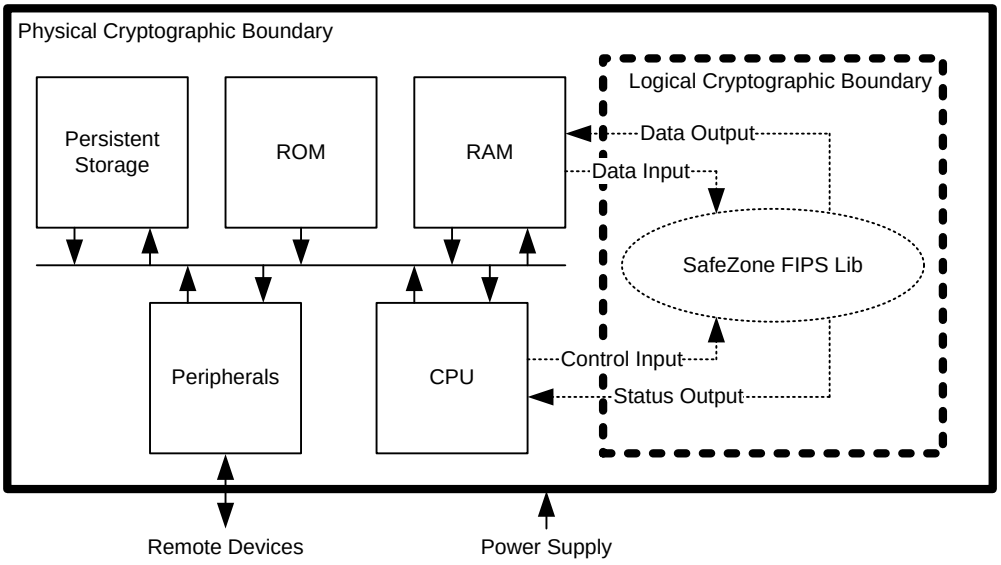
1 Introduction

SafeZone FIPS Cryptographic Module is a FIPS 140-2 Security Level 1 validated software cryptographic module from AuthenTec/INSIDE Secure. 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 AuthenTec/INSIDE Secure 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` 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 (v1.0.3) has been tested for validation on the following platforms:

Processor / Tested Platform	Operating System
ARM Cortex-A9 (ARMv7) / Pandaboard	Linux[1]/kernel 2.6 (Ubuntu 11.04) (single-user mode)
ARM Cortex-A9 (ARMv7) / Pandaboard	Android/2.3 (single-user mode)
ARM Cortex-A9 (ARMv7) / Pandaboard	Android/4.0 (single-user mode)

Compliance is maintained for all of the above operating system platforms on which the binary executes unchanged. The module has been confirmed by the vendor to be operational on the following platforms, for which the module can be recompiled according to FIPS 140-2 Implementation Guidance G.5.

Implementation Guidance G.5 Recompilation	
Processor	**Operating System**
ARM Cortex-M3 (ARMv7)	FreeRTOS (single-user mode)

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, FIPS-compliant manner.

1.2 Security level

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

[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.

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
DRM	Digital Rights Management
DSS	Digital Signature Standard
EC	Elliptic Curve
FIPS	Federal Information Processing Standard
KEM	Key-Encapsulation Mechanism (See NIST SP 800-56B)
SHS	Secure Hash Standard
SRDI	Security Relevant Data Item
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`).
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 operator must not call `ptrace` or `strace` functions, or run `gdb` or other debugger when the module is in the FIPS mode.
3. 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.
4. 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.
5. 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.
6. 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.
7. The SafeZone FIPS Cryptographic Module aims to be flexible and therefore it includes support for cryptographic algorithms or key lengths that are considered secure only until 2013 according NIST SP 800-131A. It is the responsibility of the SafeZone FIPS Cryptographic Module user to ensure that algorithms or key lengths are not used anymore once they are deprecated.
8. 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 1024 bits, 2048 bits and 3072 bits. It is not permissible to generate keys using other RSA modulus sizes.
    b. For RSA Signature Generation using modulus sizes 1536 bits or 4096 bits the RSA private key must be provided by the operator, and RSA Key Pair Generation must not be used.

9. 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) 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 must be initialized using the FL_LibInit API function and it must be ensured that the FL_LibInit returns FLR_OK (constant value 0), which signifies a successful module initialization.

The SafeZone FIPS Cryptographic Module does not support operator authentication and thus does not require any authentication itself. The SafeZone FIPS Cryptographic Module is always in FIPS-approved mode once initialized and thus no action is required to pick the mode of operation either. Usually, the module does not require any special set-up or initialization except for installation.

The module is designed to be used only in FIPS-approved mode and does not provide functions for initialization in non-FIPS mode.

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` or `FL_AssetAllocateBasic`. 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
RSA FIPS 186-2	1024, 1536, 2048, 3072, and 4096 bit keys; PKCS #1 v1.5 and PSS	#1061
RSA FIPS 186-3	1024, 2048, and 3072 bit keys; PKCS #1 v1.5 and PSS	#1061
DSA FIPS 186-3	P=1024/N=160, P=2048/N=224, P=2048/N=256, P=3072/N=256	#648
ECDSA FIPS 186-2/3	NIST P-192, P-224, P-256, P-384 and P-521 curves	#299
AES FIPS 197, NIST SP 800-38A	128, 192, 256 bit keys; ECB, CBC, CTR mode	#2041
AES CCM NIST SP 800-38C	128, 192, 256 bit keys	#2041
AES GCM NIST SP 800-38D	128, 192, 256 bit keys	#2041
XTS-AES NIST SP 800-38E	256, 512 bit keys (128-bit or 256-bit encryption strength)	#2041
Triple-DES NIST SP 800-67	192 bit keys; ECB and CBC mode	#1318
CMAC NIST SP 800-38B	128, 192, 256 bit keys	#2041

Algorithm	Implementation Details	Algorithm Certificate
HMAC FIPS 198-1	80-512 bit keys; SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	#1240
SHS FIPS 180-3	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512; BYTE only	#1787
DRBG NIST SP 800-90	AES-256-CTR with df	#203
KEM NIST SP 800-56B	1024, 1536, 2048, 3072, and 4096 bit keys; RSA-KEM-KWS-basic; vendor affirmed; key-wrapping; key establishment methodology provides between 80 and 150 bits of encryption strength	N/A, Vendor-affirmed
OAEP NIST SP 800-56B	1024, 1536, 2048, 3072, and 4096 bit keys; RSA-OAEP; vendor affirmed; key-wrapping; key establishment methodology provides between 80 and 150 bits of encryption strength	N/A, Vendor-affirmed
PBKDF2 NIST SP 800-132	with SHA-1, SHA-256	N/A, Vendor-affirmed
KDF NIST SP 800-108	80-512 bit keys; SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, AES-CMAC; vendor affirmed; counter, feedback and double pipeline modes	N/A, Vendor-affirmed Key derivation methodology provides between 80 and 256 bits of encryption strength.
TLS-PRF NIST SP 800-135	Key Derivation	N/A, Vendor-affirmed
FFC Diffie-Hellman primitive; A part of NIST SP 800-56A	Key Agreement Primitives; 1024, 2048, 3072 bit modular Diffie-Hellman groups	CVL: #21 Key establishment methodology provides between 80 and 128 bits of encryption strength.

Algorithm	Implementation Details	Algorithm Certificate
ECC CDH primitive; A part of NIST SP 800-56A	Key Agreement Primitives; NIST P-192, P-224, P-256, P-384 and P-521 curves	CVL: #21 Key establishment methodology provides between 80 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; 1024, 1536, 2048, 3072, and 4096 bit keys	(RSA Cert. #1061) Key establishment methodology provides between 80 and 150 bits of encryption strength.
AES Key Wrap	Key Wrapping 128, 192, 256 bit keys	(AES Cert. #2041) Key establishment methodology provides between 128 and 256 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	The 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. AuthenTec/INSIDE Secure 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 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	196 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
KDF Key Derivation key	NIST SP 800-108 + HMAC or CMAC	80-512 bits	Key created for the purpose of deriving other keys as specified in NIST SP 800-108.	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
TLS-PRF Key Derivation Key	NIST SP 800-135	80-512 bits	Key created for the purpose of key derivation using TLS1.0 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	80-512 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	80-512 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)	1024, 1536, 2048, 3072, 4096 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=1024/N=160, 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-192, 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 key wrapping and unwrapping using AES Key Wrap algorithm	Crypto Officer, User	Plaintext in RAM	Power-off, FL_AssetFree, FL_LibUnInit
Diffie-Hellman Private Value	Diffie-Hellman	P=1024/N=160, 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
EC Diffie-Hellman Private Value	EC Diffie-Hellman	P-192, 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

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
KEM Unwrapping Key	RSA Private Key (CRT)	1024, 1536, 2048, 3072, 4096 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
OAEP Unwrapping Key	RSA Private Key (CRT)	1024, 1536, 2048, 3072, 4096 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
RSA Unwrapping Key	RSA Private Key (CRT)	1024, 1536, 2048, 3072, 4096 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 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 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

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
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, 1536, 2048, 3072, 4096 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
Diffie-Hellman Public Value	Diffie-Hellman	P=1024/N=160, 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-192, 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
KEM Wrapping Key	RSA Public Key	1024, 1536, 2048, 3072, 4096 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

ID	Algorithm	Size	Description	Origin	Storage	Zeroization Method
OAEP Wrapping Key	RSA Public Key	1024, 1536, 2048, 3072, 4096 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
RSA Wrapping Key	RSA Public Key	1024, 1536, 2048, 3072, 4096 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.

5.3 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 three following tables: for secret keys, private keys, and public keys. 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																	
Show Status (FL_LibStatus)																	
Zeroize (FL_LibUnInit)		D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
On-demand self-test (FL_LibSelfTest)																	
Create Key (FL_AssetAllocate, FL_AssetAllocateBasic, FL_AssetLoadValue)		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											
Authenticated Encryption/Decryption with Associated Data (FL_EncryptAuthInitRandom, FL_EncryptAuthInitDeterministic, FL_CryptAuthInit[2], 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					
Digest Generation (FL_HashInit, FL_HashContinue, FL_HashFinish, FL_HashFinishKeep, FL_HashSingle)																	
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				
AES Key Wrapping (FL_AssetsWrapAes)		R	R	R	R	R	R	R	R	R	R	R	XR				

[2] Function may only be used for AES-CCM encryption, in particular the function shall not be used for 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																	
AES Key Unwrapping (FL_AssetsUnwrapAes)		W	W	W	W	W	W	W	W	W	W	W	XW				
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	
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)																	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	KEM Unwrapping Key	OAEP Unwrapping Key	RSA Unwrapping Key	DRBG state: Key / V
Role/Service										
User role or Crypto Officer Role										
Show Status (FL_LibStatus)										
Zeroize (FL_LibUnInit)		D	D	D	D	D	D	D	D	D
On-demand self-test (FL_LibSelfTest)										
Create Key (FL_AssetAllocate, FL_AssetAllocateBasic, FL_AssetLoadValue)		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)										XW
Generate Key Pair (FL_AssetGenerateKeyPair)		W	W	W	W	W	W	W	W	XW

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	KEM Unwrapping Key	OAEP Unwrapping Key	RSA Unwrapping Key	DRBG state: Key / V
Role/Service										
DSA/Diffie-Hellman Domain Parameter and Key Pair Generation (FL_AssetGenerateKeyPair)			W		W					XW
Signature Generation (FL_HashSignFips186, FL_HashSignPkcs1Pss)		X	X	X						XW
AES Key Wrapping (FL_AssetsWrapAes)		R	R	R	R	R	R	R	R	
AES Key Unwrapping (FL_AssetsUnwrapAes)		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**	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	KEM Wrapping Key	OAEP Wrapping Key	RSA Wrapping Key	DRBG state: Key / V
Role/Service											
User role or Crypto-Officer Role											
Show Status (FL_LibStatus)											
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_AssetLoadValue)			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	XW
DSA/Diffie-Hellman Domain Parameter and Key Pair Generation (FL_AssetGenerateKeyPair)				W		W					XW
Public Key Validation (FL_AssetCheck)			X	X	X	X	X	X	X	X	

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	KEM Wrapping Key	OAEP Wrapping Key	RSA Wrapping Key	DRBG state: Key / V
Role/Service											
DSA/Diffie-Hellman Domain Parameter Verification (FL_AssetCheck)				X		X					
Signature Verification (FL_HashVerifyFips186, 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)		X									XW

5.4 Algorithm details

Some of the FIPS Publications or NIST Special Publications require that the Cryptographic Module Security Policy mentions important configuration items for those algorithms.

5.4.1 NIST SP 800-108: Key Derivation Functions

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

5.4.2 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.

5.4.3 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 the `FL_CryptAuthInit` function for encryption with AES GCM. The `FL_CryptAuthInit` function is still used for decryption.

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.

5.4.4 NIST SP 800-90: Deterministic Random Bit Generator

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. If Crypto Officer uses `/dev/random` 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.

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 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 when the SafeZone FIPS Cryptographic Module is initialized with the `FL_LibInit` API function.

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`.

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 does not mitigate against any specific attacks outside the scope of FIPS 140-2.