Digital.ai
Key & Data Protection Module

FIPS 140-3 Non-Proprietary Security Policy

Version *1.0*

TABLE OF CONTENTS

TABLE OF TABLES

1. GENERAL INFORMATION

1.1 OVERVIEW

This document defines the Security Policy for the *Digital.ai Key & Data Protection Module* software cryptographic module. The module is a sophisticated implementation of whitebox cryptography (Class A safety-critical software per IEC 62304) that ingests cryptographic keys for use in the cryptographic operations performed by methods in the module's libraries. Whitebox cryptography is a field of study and a set of techniques aimed at protecting cryptographic algorithms and keys from being compromised in a hostile environment where an attacker has complete access to the implementation and execution of the cryptographic algorithms. The module is designed to allow a user to use both obfuscation and encryption on sensitive data and chain together to reduce or remove the possibility of a successful attack. The module meets FIPS 140-3 overall Security Level 1 requirements and is intended for use by US Federal agencies and other markets that require FIPS 140-3 validation. The cryptographic boundary of the module is the entire monolithic library file.

1.2 SECURITY LEVELS

The module meets the overall requirements of FIPS 140-3 Security Level 1.

Table 1 – Security Levels

ISO/IEC 24759 Section 6	FIPS 140-3 Section Title	Security Level
1	General	1
2	Cryptographic Module Specification	1
3	Cryptographic Module Interfaces	1
4	Roles, Services, and Authentication	1
5	Software/Firmware Security	1
6	Operational Environment	1
7	Physical Security	N/A
8	Non-Invasive Security	N/A
9	Sensitive Security Parameter Management	1
10	Self-Tests	1
11	Life-Cycle Assurance	1
12	Mitigation of Other Attacks	N/A
	Overall Level	1

2. CRYPTOGRAPHIC MODULE SPECIFICATION

2.1 DESCRIPTION

Purpose and Use: The module is a sophisticated implementation of whitebox cryptography that ingests keys for use in the cryptographic operations performed by methods in the *Digital.ai Key & Data Protection Module* libraries.

Module Type: The module is defined as a software module (*refer to ISO/IEC 19790, Section 7.2.2*)

Module Embodiment: The module and OE are defined as a multi-chip standalone module.

Module Characteristics: The module comprises a single, dynamic library built as an object file (***libtfit_fips_module.so***).

Cryptographic Boundary: The cryptographic boundary is defined as the *Digital.ai Key & Data Protection Module*. The boundary encompasses the entire monolithic library file named *'libtfit_fips_module.so'.*

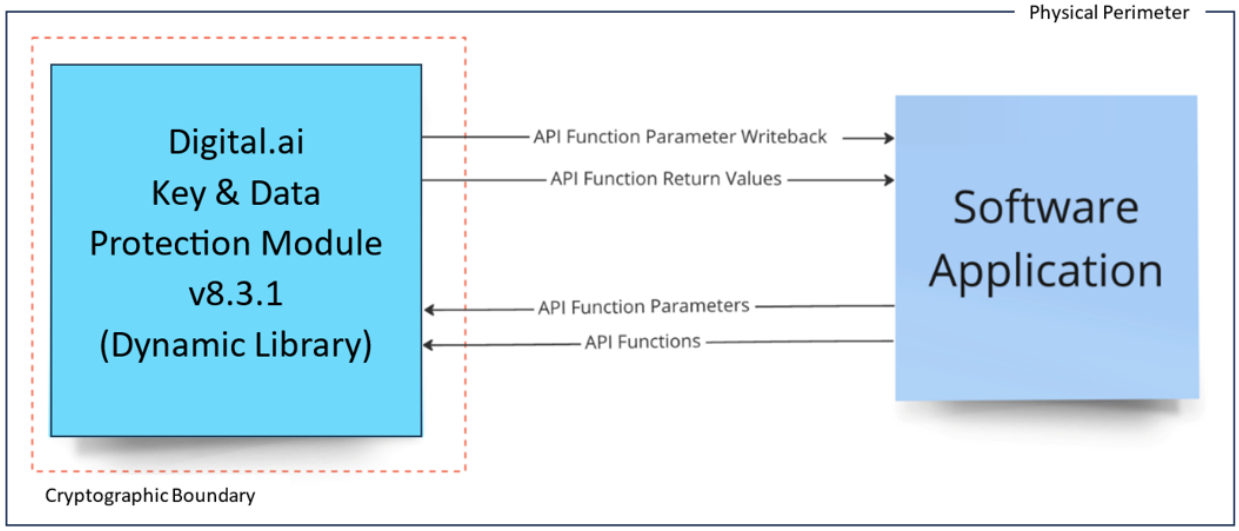

Figure 1 – Cryptographic Boundary

Figure 1 describes the module operational environment. The boundary is the entire monolithic library files. The logical interface is defined as the API for the library.

The *Digital.ai Key & Data Protection Module* is loaded as a dynamic library as part of a software application.

2.2 VERSION INFORMATION

The *Digital.ai Key & Data Protection Module* is a software cryptographic module developed to meet the requirements of FIPS 140-3 Security Level 1 (*refer to Table 1*).

Table 2 – Version Information

Module	Software Version	Operational Environment (OE)
Digital.ai Key & Data Protection Module	8.3.1	Android 64 bit (Android v12 on ARMv8.2-A Cortex-A75)

2.3 Operating Environment (OE)

2.3.1 Software, Firmware, Hybrid Tested Operating Environment

The module's operational environment (OE) is defined as modifiable and includes all the module's components, the hardware platform, and the operating system.

Table 3 – Operating Environment

Operating System	Hardware Platform	Processor(s)	PAA/Acceleration
Android v12	N/A	ARMv8.2-A Cortex-A75	N/A

2.3.2 Executable Code Sets

The module comprises a dynamic library in the form of an object file that can in turn be linked with an executable overarching application.

Table 4 – Code Sets

Package/File Names	Software Version	Integrity Test
libtfit_fips_module.so	8.3.1	HMAC-SHA-256

2.3.3 Vendor Affirmed Operating Environments

The module does not support any vendor affirmed operating environments.

2.4 Excluded Components

The module does not exclude any components from the requirements of FIPS 140-3.

2.5 Modes of Operation

The *Digital.ai Key & Data Protection Module* only has an 'approved' mode of operation. There are no undefined or 'non-approved' modes or services within the module. By default, the approved mode is entered into when powering on the module. When called successfully, each of the approved services returns a '0' integer value. If any other value is returned, the module transitions to its defined error state. The module returns '*Digital.ai Key & Data Protection Module Version: 8.3.1*' to indicate its approved mode of operation.

N.B. The module does not incorporate a degraded mode of operation (*refer to ISO/IEC 19790 Section 7.2.4.3).*

2.6 Security Functions

2.6.1 Approved Algorithms

The module implements the following approved cryptographic functions listed in the following table:

Table 5 – Approved Algorithms

CAVP Cert(s)	Algorithm	Standard	Mode /Method	Key Size	Use/Function
A3516	AES	FIPS 197, NIST SP 800-38A	ECB	256-bit	Encryption / Decryption
A3516	ECDSA	FIPS 186-4	Sig Ver	P-256	Signature Verification
A3516	HMAC-SHA-256	FIPS 198-1	N/A	256-bit	Software Integrity Check
A3516	KAS-ECC CDH Component	SP 800-56Ar3	ECC CDH	P-256	Calculates a Shared Secret (Z) value
A3516	SHS	FIPS 180-4	SHA-256, SHA-512	N/A	Digital Signature Verification Hash Generation

2.6.2 Vendor Affirmed Algorithms

There are no vendor affirmed algorithms within boundaries of the module.

Table 6 – Vendor Affirmed Algorithms

Algorithm	Caveat	Use/Function
N/A	N/A	N/A

2.6.3 Non-Approved Algorithms

There are no non-approved allowed algorithms within boundaries of the module.

Table 7 – Non-Approved Allowed Algorithms

Algorithm	Caveat	Use/Function
N/A	N/A	N/A

2.6.4 Non-Approved, Allowed Algorithms with No Security Claimed

There are no non-approved algorithms within boundaries of the module.

Table 8 – Non-Approved Allowed with No Security Claimed

Algorithm	Caveat	Use/Function
N/A	N/A	N/A

2.6.5 NON-APPROVED

There are no non-approved not allowed algorithms within boundaries of the module.

Table 9 – Non-Approved Not Allowed Algorithms

Algorithm	Caveat	Use/Function
N/A	N/A	N/A

2.7 SECURITY FUNCTION IMPLEMENTATIONS (SFI)

Table 10 – Security function implementation

Name	Type	Description	SF Properties	Algorithms / CAVP Cert.
ECDH	KAS-ECC CDH-Component	Shared Secret Computation	P-256 curve	KAS-ECC CDH-Component / A3516

2.8 ALGORITHM SPECIFIC INFORMATION

The module utilizes only approved algorithms that are tested and validated under the Cryptographic Algorithm Validation Program (CAVP):

- AES per FIPS 197
- ECDSA per FIPS 186-4
- HMAC per FIPS 198-1
- KAS-ECC CDH-Component per NIST SP800-56Ar3
- SHA-256 and SHA-512 per FIPS 180-4

2.9 RNG AND ENTROPY

The module does not support RNG or entropy generation.

2.10 KEY GENERATION

The module does not support the generation of cryptographic keys. The module does not implement random number generation.

2.11 KEY ESTABLISHMENT

The module does not support key establishment methods.

2.12 INDUSTRY PROTOCOLS

The module does not support any industry standard protocols.

2.13 Security Design and Rules of Operation

The module design corresponds to the module security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS 140-3 Level 1 module.

- The operator shall be capable of commanding the module to perform the power up self-tests by power cycling the system or by re-loading the library into the application.
- Power up self-tests do not require any operator action.
- Data output shall be inhibited by self-tests, zeroization and error states.
- Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
- The module does not support concurrent operators.
- The module does not support a maintenance interface or role.
- The module does not support manual key entry.
- The module does not have any external input/output devices used for entry/output of data.
- All CSPs are protected from unauthorized access, use, disclosure, modification, and substitution. The module does not persistently store CSPs. CSPs are provided to the module for the purpose of executing cryptographic operation.
- All PSPs are protected from unauthorized modification and substitution.
- The module does not output cryptographic keys.
- The operator can command the module to perform the Pre-Operational and Conditional self-tests at any time by loading the module in memory space of an application requiring its services.
- When the module is in an error state, the operator shall not have access to any cryptographic service.

2.14 Initialization

The module does not require specific initialization as no CSPs are stored within its cryptographic boundary. The module is loaded into an application address space requiring its services. On load, the module immediately performs self-testing of all its cryptographic functions and output status of the tests (*refer to Section 10*).

3. CRYPTOGRAPHIC MODULE INTERFACES

3.1 PORTS & INTERFACES

Table 11 – Ports and Interface

Physical Port	Logical Interface	Data Passed over the Interface
N/A	Control input	Defined parameters used to configure the behavior of the module's specific services. Externally visible *Digital.ai Key & Data Protection Module* library symbols (module API Controls).
N/A	Control output	N/A
N/A	Data output	Data returned in which is the result of the API service transaction.
N/A	Data input	Data passed in which the API service will be transacted upon.
N/A	Status output	API service parameter return values.

4. ROLES, SERVICES, AND AUTHENTICATION

4.1 AUTHENTICATION METHODS

The module does not implement any authentication methods.

4.2 ROLES

The module supports a single, Cryptographic Officer (CO) role. The role is assumed implicitly and has access to all services. No other roles are supported.

The module does not support a maintenance role or bypass capability. The module does not support concurrent operators.

Table 12 – Roles, Services, Input & Output

Role	Service	Input	Output
Cryptographic Officer (CO)	AES Encrypt	Plaintext data to encrypt	Status & Ciphertext Data
	AES Decrypt	Ciphertext data to decrypt	Status & Plaintext Data
	ECDSA Verify	Plaintext data and digital signature to be verified	Status
	Generate Shared Secret	Other Party's Public Key	Status & Shared Secret (Z)
	Zeroize	N/A	Status
	Create Hash	Data to be Hashed, length, & algorithm	Status, digest, & length

Role	Service	Input	Output
	Generate HMAC	Data to be input into HMAC algorithm	Status & MAC
	Verify HMAC	Data to be input into HMAC algorithm & HMAC to be verified	Status
	Load AES Encryption Key	AES Key	Status
	Load AES Decryption Key	AES Key	Status
	Load ECDH Private Key	ECDH Priv Key	Status
	Load ECDSA Public Key	ECDSA Pub Key	Status
	Load HMAC Generate Key	HMAC Key	Status
	Load HMAC Verification Key	HMAC Key	Status
	Get Status	N/A	Status
	Show Version	N/A	Module name & Version number
	Self-Test	N/A	N/A

Table 13 – Roles

Role	Authentication Method	Authentication Strength
Cryptographic Officer	N/A	N/A

4.3 Approved Services

All services implemented by the module are listed in the table below. Each service description also describes all usage of CSPs by the service.

Table 14 – Approved Services

Service	Description	Approved Security Functions	Keys & SSPs	Role	Access Rights to Keys and / or SSPs	Indicator
AES Encrypt	AES encryption	AES ECB Encrypt	AES Key	CO	AES Key: **E**	Integer value (0 = Success or Error Code)
AES Decrypt	AES decryption	AES ECB Decrypt	AES Key	CO	AES Key: **E**	Integer value (0 = Success or Error Code)
ECDSA Verify	ECDSA digital signature verification	ECDSA Sig Ver	ECDSA Pub Key	CO	ECDSA Pub Key: **E**	Integer value (0 = Verified or Error Code)
Generate Shared Secret	Calculates a shared secret	KAS-ECC CDH-Component	ECDH Pub Key ECDH Priv Shared Secret (Z)	CO	ECDH Pub Key: **W, E, Z** ECDH Priv: **E, Z** Shared Secret (Z): **G**	Integer value (0 = Success or Error Code)
Zeroize	Zeroizes all SSPs	N/A	All SSPs	CO	All SSPs: **Z**	Integer value (0 = Success or Error Code)

Service	Description	Approved Security Functions	Keys & SSPs	Role	Access Rights to Keys and / or SSPs	Indicator
Create Hash	Generates a SHA digest	SHA-256 or SHA-512	N/A	CO	N/A	Integer value (0 = Success or Error Code)
Generate HMAC	Generates HMAC	HMAC-SHA-256	HMAC Key	CO	HMAC Key: **E**	Integer value (0 = Success or Error Code)
Verify HMAC	Verifies HMAC	HMAC-SHA-256	HMAC Key	CO	HMAC Key: **E**	Integer value (0 = Verified or Error Code)
Load AES Encryption Key	Loads the AES Key	N/A	AES Key	CO	AES Key: **W**	Integer value (0 = Success or Error Code)
Load AES Decryption Key	Loads the AES Key	N/A	AES Key	CO	AES Key: **W**	Integer value (0 = Success or Error Code)
Load ECDH Private Key	Loads the EC private key for use in the shared secret calculation.	N/A	ECDH Priv Key	CO	ECDH Priv Key: **W**	Integer value (0 = Success or Error Code)
Load ECDSA Public Key	Loads the ECDSA Pub Key	N/A	ECDSA Pub Key	CO	ECDSA Pub Key: **W**	Integer value (0 = Success or Error Code)
Load HMAC Generate Key	Loads the HMAC Key	N/A	HMAC Key	CO	HMAC Key: **W**	Integer value (0 = Success or Error Code)
Load HMAC Verification Key	Loads the HMAC Key	N/A	HMAC Key	CO	HMAC Key: **W**	Integer value (0 = Success or Error Code)
Get Status	Returns the status of the module	N/A	N/A	CO	N/A	Integer value (0 = Success or Error Code)
Show Version	Returns the ID and version of the module	N/A	N/A	CO	N/A	Module name & version
Self-test	Self-test service is invoked by loading or reloading the module	All	N/A	CO	N/A	<Self-Test Name> + <Passed or Failed>

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

Non-Approved Services

The module does not implement any non-Approved services.

4.4 External Software Loaded

The module does not implement external software loading capabilities.

5. SOFTWARE/FIRMWARE SECURITY

The executable code is provided as a dynamic library (***libtfit_fips_module.so***) to a consuming application which interfaces to the library. The library's integrity is protected using HMAC-SHA-256 at the time of development.

The Conditional and Pre-Operational Self-Tests run upon instantiation of the module. All cryptographic algorithm self-tests (CAST) must successfully pass its applicable self-test for the module to transition to its 'normal' operation.

If a Self-Test fails for any reason, a message is printed to standard error (the assert logcat on Android) and forces termination of the application.

5.1 INTEGRITY TECHNIQUES

As part of the pre-operational self-tests, a software integrity test comprising HMAC-SHA-256 is performed. If the software integrity test fails, the module does not load and returns an error over stderr (*refer to Section 10.1*).

5.2 INITIATE ON DEMAND

The module's consuming application can be restarted at any time to perform a periodic self-test.

6. OPERATIONAL ENVIRONMENT

The module's operational environment supports process isolation. Each process is logically separated from all other processes by the operating system and hardware. The module functions entirely within a single process. The operational environment enforces process separation using virtual memory. The virtual memory system uses the memory management unit (MMU) hardware to prevent processes from accessing the memory image of another process.

6.1 OPERATIONAL ENVIRONMENT TYPE AND REQUIREMENTS

The module is designed to run on an Android v12 operational environment (OE). The OE is classified as modifiable. The tested OE and platforms are detailed in Table 2. The module has no interaction with the OE, so all requirements for FIPS 140-3 Security Level 1 are satisfied.

6.2 CONFIGURATION SETTINGS AND RESTRICTION

There are no security rules, settings, or restrictions on the configuration of the operational environment imposed by the module. The module has no interaction with the operational environment.

The consuming application may define security rules, configuration settings, or restrictions on the OE.

7. PHYSICAL SECURITY

Physical Security requirements are not applicable, as the module is a FIPS 140-3 Security Level 1 software module.

8. NON-INVASIVE SECURITY

The module does not claim any security from non-invasive attack methods.

9. SENSITIVE SECURITY PARAMETER (SSP) MANAGEMENT

The module incorporates both Critical Security Parameters (CSPs) and Public Security Parameters (PSPs).

9.1 SSPs

The module incorporates SSPs as defined with Table 15.

Table 15 – SSPs

Key/CSP Name	Strength	Security Function & Cert. Number	Generation	Import / Export	Establishment	Storage	Zeroization	Use & related SSPs
AES Key	256 bits	AES-ECB (Cert. #A3516)	N/A	Entry: Plaintext Output: N/A	N/A	RAM	'Zeroize' Service and immediately after use	Data Encryption / Decryption
ECDH Priv Key	128 bits	ECC CDH Component (Cert. #A3516)	N/A	Entry: Plaintext Output: N/A	N/A	RAM	'Zeroize' Service and immediately after use	Generate Shared Secret (Z)
ECDH Pub Key	128 bits	ECC CDH Component (Cert. #A3516)	N/A	Entry: Plaintext Output: N/A	N/A	RAM	'Zeroize' Service and immediately after use	Generate Shared Secret (Z)
ECDSA Pub Key	128 bits	ECDSA Sig Ver (Cert. #A3516)	N/A	Entry: Plaintext Output: N/A	N/A	RAM	'Zeroize' Service and immediately after use	ECDSA Digital Signature Verification
HMAC Key	256-bits	HMAC-SHA-256 (Cert. #A3516)	N/A	Entry: Plaintext Output: N/A	N/A	RAM	'Zeroize' Service and immediately after use	Generate and Verify HMACs
Shared Secret (Z)	256 bits	ECC CDH Component (Cert. #A3516)	N/A	Entry: N/A Output: Plaintext	N/A	RAM	'Zeroize' Service and immediately after use	Not used by the module

9.2 ZEROIZATION

SSPs are not persistently stored. During normal operation, the module explicitly erases copies of SSPs in volatile memory (e.g., RAM) by overwriting with zeros after their use. SSPs temporarily stored in volatile memory can be zeroized via the 'Zeroize' service.

10. SELF-TESTS

On instantiating the consuming application, the module performs the self-tests described in Tables 16 & 17 below. All KATs must be completed successfully prior to any use of cryptography by the module. If one of the KATs fails, the module transitions to its appropriate error state.

10.1 PRE-OPERATIONAL SELF-TESTS

The module implements both preoperational and conditional self-tests. Since the pre-operational self-test for this module comprises the software integrity test utilizing the HMAC-SHA-256 algorithm, the HMAC-SHA-256 KAT for the conditional self-test is invoked prior to the software integrity test.

Table 16 – Pre-Operational Self-Tests

Algorithm	Implementation	Test Properties	Test Method	Type	Indicator	Details
HMAC-SHA-256	Software Integrity Test	256-bit	KAT	Software Integrity	"HMAC/SHA2_256 checksum passed" or "HMAC/SHA2_256 checksum failed"	Verifies the HMAC-SHA-256 message authentication code for the software.

10.2 CONDITIONAL SELF-TESTS

Table 17 – Conditional Self-Tests

Algorithm Test	Implementation	Test Properties	Test Method	Type	Indicator	Test Details	Condition
AES-ECB Encrypt	Software	256-bit key	KAT	CAST	"aes_KAT passed" or "aes_KAT failed"	Encrypt	Module load
AES-ECB Decrypt	Software	256-bit key	KAT	CAST	"aes_KAT passed" or "aes_KAT failed"	Decrypt	Module load

ECC CDH Component	Software	P-256	KAT	CAST	"ecdh_KAT passed" or "ecdh_KAT failed"	Primitive Shared Secret ('Z') Computation KAT	Module load
ECDSA Sig Ver	Software	P-256	KAT	CAST	"ecdsa_KAT passed" or "ecdsa_KAT failed"	Verify	Module load
HMAC	Software	256-bit key	KAT	CAST	"hmac_KAT passed" or "hmac_KAT failed"	Generation	Module load
SHA	Software	SHA-256	KAT	CAST	sha_KAT passed or "sha_KAT failed"	Hash	Module load

10.3 Periodic Self-Tests

The module is designed to meet the requirements of FIPS 140-3 Security Level 1. It does not implement automated periodic self-testing. However, the module's consuming application may be restarted at any time to perform a periodic self-test.

10.4 Error States

The module supports the following error states.

Table 18 – Error States

State Name	Description	Conditions	Recovery Method	Indicator
Soft Error	Recoverable errors from the normal operation of the module	Error conditions from the result of a service call	The module will continue to operate as normal.	Integer value is returned from the service for handling by the calling application.
Hard Error	A fatal error that ends the current operation of the module.	Failure of pre-operational or conditional self-tests	The module aborts service, outputs error indicator, and forces a segmentation fault. The module must be restarted.	Error message is output on stderr (standard error)

11. LIFE-CYCLE ASSURANCE

Module code and documentation is stored and managed within a private and secure configuration management system.

Module code utilizes semantic versioning to each release indicating major, minor or patch releases along with release notes providing guidance in the use of each iteration of module's code. The module has a semantic version number. The Git configuration management system assigned a message digest to each commit which uniquely identifies the version.

Configuration management systems retaining module code are periodically backed up using a solution that allows both full and incremental restoration.

The procedures for secure installation, initialization, startup, and operation of the module are provided in sections 11.1, 11.2, and 11.3 of this Security Policy and the module's user guidance.

A comprehensive developer guide documenting the entire API is provided. The developer guide provides all Administrator guidance. The developer guide is versioned and managed within the same secure configuration management systems.

The configuration management system is protected with user authentication measures to prevent unauthorized access. All changes can be made only by authorized individuals. An automated system is in place to notify other users when any change has been made to prevent unknown changes from being made. All changes are made on development branches in Git and are reviewed by separate people before releasing to production.

The only maintenance required is to install new versions of the module when they are made available to fix discrepancies or add features. The update procedure is the same as the installation procedure, with the new module file(s) replacing the existing ones.

Any new version of the module is outside the scope of this validation and requires a separate FIPS 140-3 validation.

The procedures for secure installation, initialization, startup, and operation of the module are provided in module's user guidance.

Additionally, a comprehensive developer guide documenting the entire API is provided. The developer guide provides all Administrator guidance. The developer guide is versioned and managed within the same secure configuration management system.

The configuration management system is protected with user authentication measures to prevent unauthorized access. All changes can be made only by authorized individuals via access rights (i.e., read/write). An automated system is in place to notify other users when any change has been made. All changes are made on development branches in Git and are reviewed by separate people before releasing to production.

The only maintenance required is to install new versions of the module when they are made available to fix discrepancies or add features. The update procedure is the same as the installation procedure, with the new module file(s) replacing the existing ones.

Any new version of the module is outside the scope of this validation and requires a separate FIPS 140-3 validation.

11.1.1 Startup Procedures

The module does not provide a non-approved mode of operation and is as such always in an approved mode once loaded into memory by the calling application and its automated self-tests are executed successfully. Since the module does not generate keys or support authentication, it does not need to be initialized.

11.1.2 SANITIZATION

The module is a software module that does not store any persistent SSPs. The module should be uninstalled from the OE when no longer needed.

11.2 ADMINISTRATOR GUIDANCE

The Crypto Officer shall configure their Android app project according to instructions detailed in the module's user guidance.

11.3 NON-ADMINISTRATOR GUIDANCE

There is no authentication component to the Module, so no further setup is required. No zeroization is required during installation.

12. MITIGATION OF OTHER ATTACKS

The module does not claim any attack mitigation beyond FIPS 140-3 Security Level 1 requirements.

13. APPENDIX A: REFERENCES

Table 19 – References

Reference	Reference Title	Author	Date
[1]	FIPS 140-3 – Security Requirements for Cryptographic Modules	NIST	2019
[2]	FIPS 180-4 – Secure Hash Standard (SHS)	NIST	2015
[3]	FIPS 186-4 – Digital Signature Standard (DSS)	NIST	2013
[4]	FIPS 197 – Advanced Encryption Standard (AES)	NIST	2023
[5]	FIPS 198-1 – The Keyed-Hash Message Authentication Code	NIST	2008
[6]	ISO/IEC 19790 - Information technology - Security techniques - Security requirements for cryptographic modules	ISO/IEC	2014
[7]	ISO/IEC 24759 - Information technology - Security techniques - Test requirements for cryptographic modules	ISO/IEC	2017
[8]	NIST SP 800-38A – Recommendation for Block Cipher Modes of Operation: Methods and Techniques	NIST	2001
[9]	NIST SP 800-56Ar3 – Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography	NIST	2020
[10]	NIST SP 800-131Ar2 – Transitioning the Use of Cryptographic Algorithms and Key Lengths	NIST	2019
[11]	NIST SP 800-140B – CMVP Security Policy Requirements: CMVP Validation Authority Updates to ISO/IEC 24759 and ISO/IEC 19790 Annex B	NIST	2020

14. APPENDIX B: ABBREVIATIONS AND DEFINITIONS

Table 20 – Abbreviations and Definitions

Term	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CAST	Cryptographic Algorithm Self-Test
CAVP	Cryptographic Algorithm Validation Program
CO	Cryptographic Officer
ECB	Electronic Code Book
ECC CDH	Elliptic Curve Cryptography Cofactor Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EFP/EFT	Environment Failure Protection/Environment Failure Testing
FIPS	Federal Information Processing Standards
HMAC	Hashed or Hash-based Message Authentication Code
IEC	International Electrotechnical Commission
IG	Implementation Guidance
ISO	International Standards Organization
KAS	Key Agreement Scheme
KAT	Known Answer Test
OE	Operating Environment
SHS	Secure Hash Standard
SP	Security Policy
SSP	Sensitive Security Parameter
TOEPP	Tested Operational Environment's Physical Perimeter