Titaniam, Inc.

Titaniam Core Engine FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.0
Module Version 1.0
Revision Date 04/18/2022

Author	Date	Change
Titaniam	05/04/2021	Initial Release
Titaniam	06/08/2021	Updated certificate number
Titaniam	03/07/2022	NIST Comments
Titaniam	04/18/2022	NIST Comments

Table of Contents

1. Introduction

This document defines the Security Policy for the Titaniam Core Engine FIPS Cryptographic Module, hereafter denoted the Module. The Module provides a cryptographic library. The Module meets The Cryptographic Module meets FIPS 140-2 overall Level 1 and Level 3 for Area 1, Cryptographic Module Specification; Area 8, EMI/EMC; and Area 10, Design Assurance. The software version of the Cryptographic Module is 1.0.

The cryptographic module was tested on the following operational environment on the general purpose computer (GPC) platform detailed below:

The cryptographic Module was Tested in the following Environment

- Physical Hardware:
    - PowerEdge T40 Server, Intel(R) Xeon(R) E-2224G CPU @ 3.50GHz
- Operating system:
    - Ubuntu 18.04.5 LTS
- Java Version:
    - Openjdk version "11.0.11"

As per FIPS 140-2 Implementation Guidance G.5, the Cryptographic Module will remain compliant with the FIPS 140-2 validation when operating on any general purpose computer (GPC) provided that:

1. No source code has been modified.
2. The GPC uses the specified single-user platform

The Cryptographic Module is intended for use by US Federal agencies and other markets that require a FIPS 140-2 validated Cryptographic Library. The Module is a software-only embodiment; the cryptographic boundary is the Java Archive (JAR) file.

The FIPS 140-2 security levels for the Module are given in Table 1 as follows:

Table 1: Security Level of Security Requirements

Security Requirement	**Security Level**
Cryptographic Module Specification	Level 3
Cryptographic Module Ports and Interfaces	Level 1
Roles, Services, and Authentication	Level 1
Finite State Model	Level 1

Physical Security	N/A
Operational Environment	Level 1
Cryptographic Key Management	Level 1
EMI/EMC	Level 3
Self-Tests	Level 1
Design Assurance	Level 3
Mitigation of Other Attacks	N/A

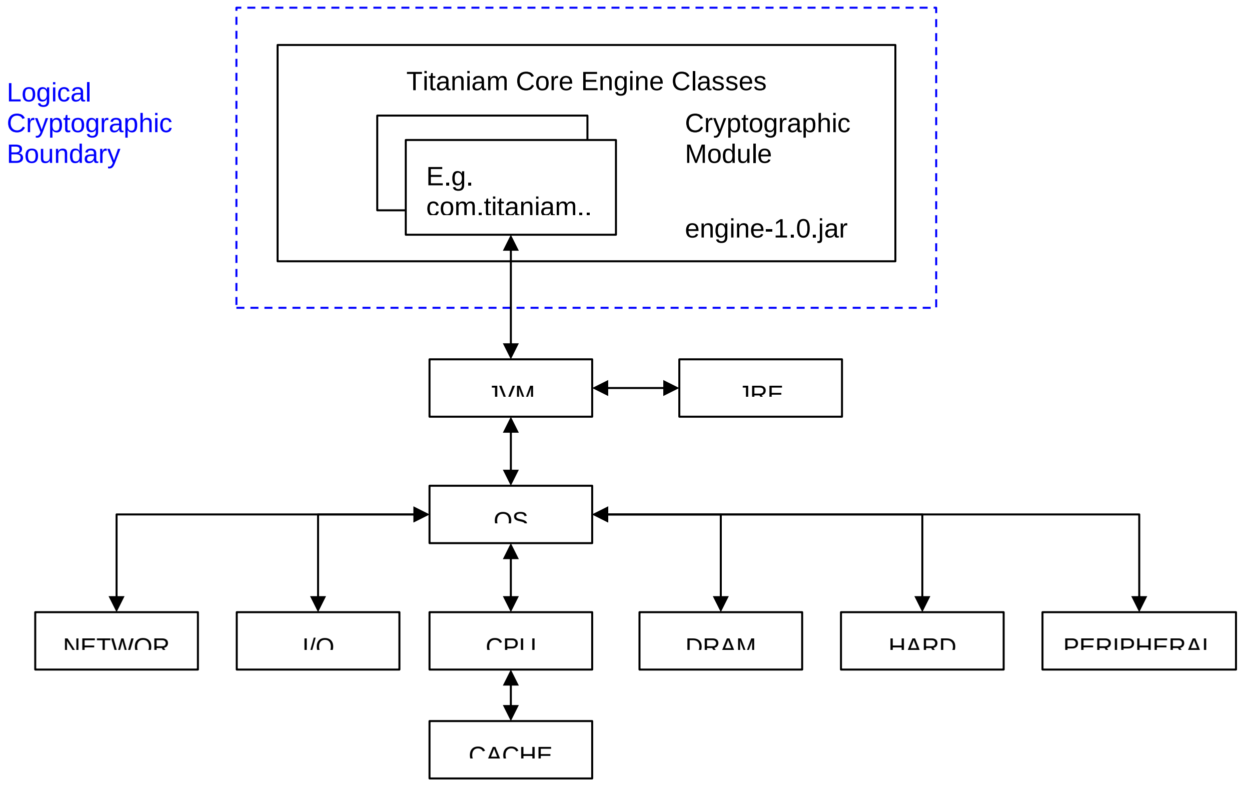
2. Logical and Physical Cryptographic Boundaries

2.1 Logical Cryptographic Boundary

The executable for the Module is: titaniam-engine.jar (/lib/titaniam-engine.jar). This module is the only software component within the Logical Cryptographic Boundary and the only software component that carries out cryptographic functions covered by FIPS 140-2.

Figure 1 shows the logical relationship of the cryptographic module to the other software and hardware components of the computer. The Titaniam classes are executed on the Java Virtual Machine (JVM) using the classes of the Java Runtime Environment (JRE). The JVM is the interface to the computer’s Operating System (OS) that is the interface to the various physical components of the computer. The physical components of the computer are discussed further in Section 6. Abbreviations introduced in Figure 1 that describe physical components are: Central Processing Unit (CPU), Dynamic Random Access Memory (DRAM) and Input Output (I/O).

Figure 1 - Block Diagram of Software for Titaniam Core Engine



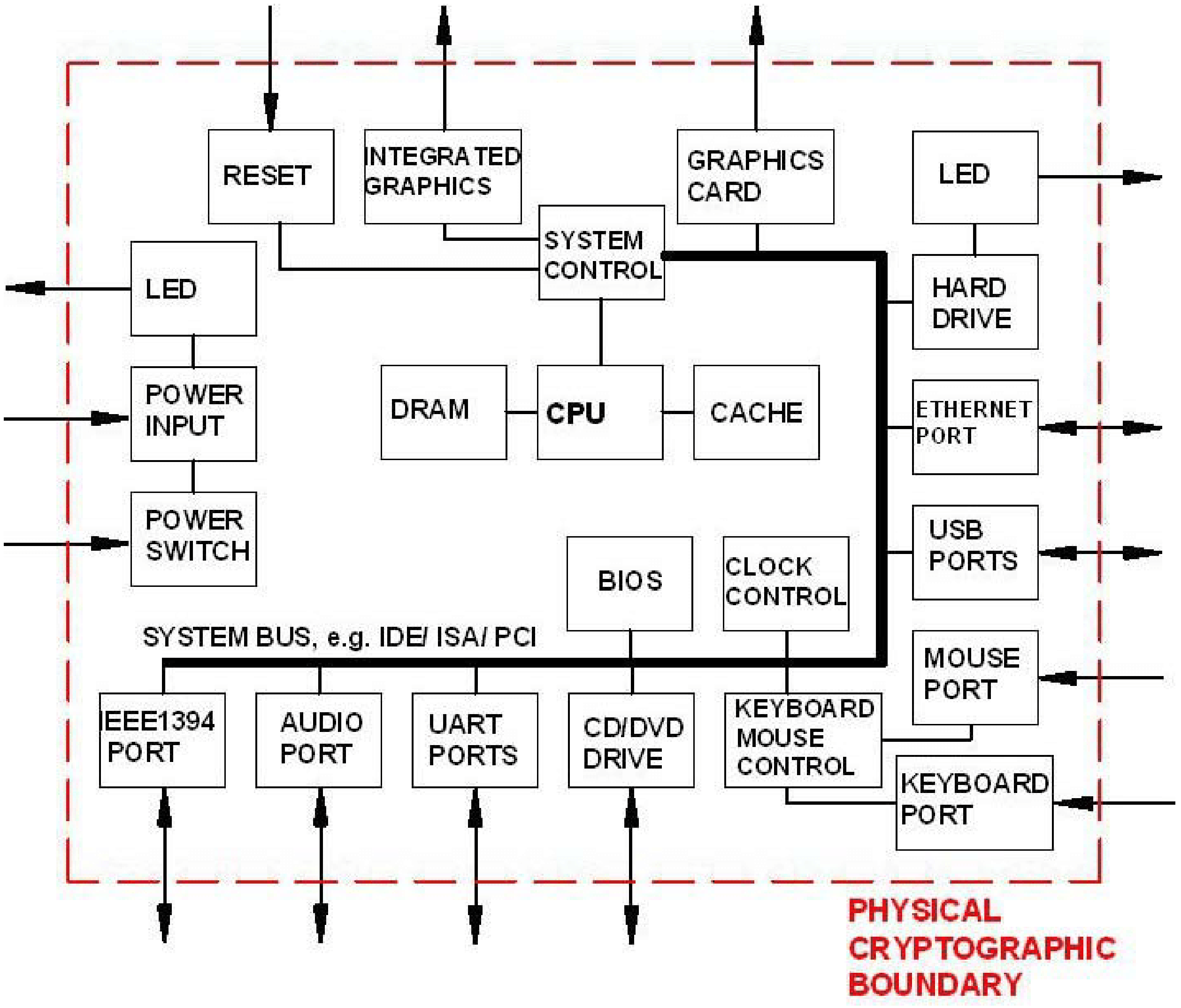
2.2 Physical Boundary

The Titaniam Core Engine Module runs on a General Purpose Computer (GPC). The Physical Cryptographic Boundary for the module is the case of that computer. Figure 2 shows a block diagram of the physical components of a typical GPC and the ports or interfaces across the Physical Cryptographic Boundary.

All the physical components are standard electronic components; there are not any custom integrated circuits or components dedicated to FIPS 140-2 related functions.

Abbreviations introduced in Figure 2 are: Basic I/O System (BIOS), Integrated Device Electronics (IDE), Institute of Electrical and Electronic Engineers (IEEE), Instruction Set Architecture (ISA), Peripheral Component Interconnect (PCI), Universal Asynchronous Receiver/Transmitter (UART) and Universal Serial Bus (USB). Input or output ports are designated by arrows with single heads, while I/O ports are indicated by bidirectional Arrows.

Figure 2 - Block Diagram of the Physical Components of a typical GPC



For FIPS 140-2 purposes, the Titaniam Core Engine Module is defined as a "multi-chip standalone module", therefore, the module's physical ports or interfaces are defined as those for the hardware of the GPC. These physical ports are separated into the logical interfaces defined by FIPS 140-2, as shown in Table 3.

The Titaniam Core Engine Module is a software-only module, and, therefore, control of the physical ports is outside of the module's scope. The module does provide a set of logical interfaces which are mapped to the following FIPS 140-2 defined logical interfaces: data input, data output, control input, status output, and power. When the module performs self-tests, if it gets into an error state, or if it is generating keys, or if it is performing zeroization, the module prevents all output on the logical data output interface as only the thread performing the operation has access to the data. The module in an error state, does not return any output data, only an error value.

Table 3: Logical Interfaces

FIPS 140-2 Logical Interface	Module Equivalent
Data Input	API input parameters – plaintext and/or ciphertext data.
Data Output	API output parameters and return values – plaintext and/or ciphertext data
Control Input	API method calls – method calls, or input parameters, that specify commands and/or control data used to control the operation of the module.
Status Output	API output parameters and return/error codes that provide status information used to indicate the state of the module.

2.3 Modes of Operation

There will be two modes of operation: Approved and Non-approved. The module will be in FIPS-approved mode when the appropriate factory is called. To verify that a module is in the Approved Mode of operation, the user can call a FIPS-approved mode status method (`FipsModeIndicator.isFipsMode()`). If the module is configured to allow approved and non-approved mode operation, setting the following JVM system property - `–Dtitaniam.fips-approved=true` and restarting it will switch to approved mode.

In FIPS-approved mode, the module will not provide non-approved algorithms, therefore, exceptions will be called if the user tries to access non-approved algorithms in the Approved Mode.

2.4 Module Configuration

In default operation the module will start with both approved and non-approved mode enabled. If the module detects that the system property `titaniam.fips-approved` is set to `true` the module will start in approved mode and non-approved mode functionality will not be available.

3. Cryptographic Functionality

The Module implements the FIPS Approved and Non-Approved but Allowed cryptographic functions listed in Table 5 to Table 7, below.

Table 5 – Approved and CAVP Validated Cryptographic Functions

CAVP Cert	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
A1388	AES	FIPS 197, SP 800-38A	CBC, ECB	128, 256	Data Encryption/ Decryption
A1491	AES-FF1	SP 800-38G		128,192,256	Data Encryption/ Decryption
A1491	SHS	FIPS 180-4	SHA-256		Digital Signature Verification
A1491	HMAC	FIPS 198-1		128	Message Authenticatio n
A1491	KBKDF	SP 800-108			Field Key Derivation

Table 6 – Non-Approved Cryptographic Functions for use in non-FIPS mode only

Algorithm	Description
Coded Keyword	Encryption/Decryption of texts
Coded Integer	Encryption/Decryption of integers
Coded Date	Encryption/Decryption of Date
Coded IP	Encryption/Decryption of IP addresses

3.1 Critical Security Parameters

All CSPs used by the Cryptographic Module are described in this section in Table 7.

Table 7 – Critical Security Parameters (CSPs)

CSP Name	Description/Usage	Generation/Storage	Zeroization
Key Encryption Key	Used to decrypt the Seed Key	Generation: N/A (Generated by the operator) Storage: RAM	Module will actively overwrite this CSP once no longer needed; Zeroized when the module is powered off.
Seed Key	Used to generate Field Level Keys	Generation: N/A (Generated by the operator) Storage: RAM	Module will actively overwrite this CSP once no longer needed; Zeroized when the module is powered off.
SP800-108 KDF Internal State	Used during Field Level Key Generation	Generation: SP800-108 KDF Storage: RAM	Module will actively overwrite this CSP upon exiting the function.
Field Level Key (Java)	Usage: AES-CBC, AES-ECB [FIPS-197, SP 800-56C, SP 800-38D, Addendum to SP 800-38A] AES (128/192/256) encrypt key[1]	Establishment: SP800-108 KDF Storage: RAM	Module will actively overwrite this CSP once no longer needed; Zeroized when the module is powered off.
Field Level Key (Idealista)	Usage: AES-FF1 [FIPS-197, SP 800-56C, SP 800-38D, Addendum to SP 800-38A]	Establishment: SP800-108 KDF Storage: RAM	Module will actively overwrite this CSP once no longer needed; Zeroized when the module is powered off.
HMAC Key	Usage: HMAC [FIPS-197, SP 800-56C, SP 800-38D, Addendum to SP 800-38A]	Establishment: SP800-108 KDF Storage: RAM	Module will actively overwrite this CSP once no longer needed; Zeroized when the module is powered off.
IntegrityChecker Key	Usage: Power up self-tests	Generation: N/A (Externally generated and hardcoded in the module)	N/A

1

		Storage: RAM	

4. Roles, Authentication and Services

4.1 Assumption of Roles

The module supports two distinct operator roles, User and Cryptographic Officer (CO). The cryptographic module implicitly maps the two roles to the services. A user is considered the owner of the thread that instantiates the module.

Table 8 lists all operator roles supported by the module. The module does not support a maintenance role and/or bypass capability. The module does not support authentication.

Table 8 – Roles Description

Role ID	Role Description	Authentication Type
CO	Cryptographic Officer – Powers on and off the module. Installs cryptographic keys.	N/A – Authentication not required for Level 1
User	User – The user of the complete API.	N/A – Authentication not required for Level 1

4.2 Services

All services implemented by the Module are listed in Table 8 below and Table 9 describes all usage of CSPs by the service.

Table 8 lists the services. The second column provides a description of each service and availability to the Cryptographic Officer and User, in columns 3 and 4, respectively.

Table 8 – Services

Service	Description	CO	User
Initialize Module and Run Self-Tests on Demand	The JRE will initialize CryptoModule that will call for self-tests on module initialization.	X	
Show Status	A user can call FipsModeIndicator.isReady() at any		X

	time to determine if the module is ready. FipsModeIndicator.isFipsMode() can be called to determine the FIPS mode of operation.		
Zeroize / Power-off	All CSPs are zeroized upon the shutdown of the module.		X
Data Encryption	Used to encrypt data.		X
Data Decryption	Used to decrypt data.		X
Keyed Message Hashing	Used to calculate data integrity codes with HMAC-SHA-256.		X
SP 800-108 KDF	(secret input) (outputs secret) Used to calculate a value suitable to be used for a secret key from an input secret and additional input.		X

Table 9 defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as:

- G = Generate: The module generates the CSP.
- R = Read: The module reads the CSP. The read access is typically performed before the module uses the CSP.
- E = Execute: The module executes using the CSP.
- W = Write: The module writes the CSP. The write access is typically performed after a CSP is imported into the module, when the module generates a CSP, or when the module overwrites an existing CSP.
- Z = Zeroize: The module zeroizes the CSP

Table 9: CSP Access Rights within Services

Service	**Seed Key**	**Key Encryption Key**	**Field Level Key (Java)**	**Field Level Key (Idealista)**	**HMAC Key**	**SP800-108 KDF Internal State**	**Integrity Checker Key**
Initialize Module and Run Self-Tests on Demand	R	R	R,W	R,W	R,W	Z	R, X
Show Status			R	R	R		

Zeroize / Power-off	Z	Z	Z	Z	Z	Z	
Data Encryptio n	R		R,W,X	R,W,X			
Data Decryptio n	R		R,W,X	R,W,X			
Keyed Message Hashing	R				R,W,X		

4.3 Self-tests

Each time the module is powered up, it tests that the cryptographic algorithms still operate correctly and that sensitive data has not been damaged. Power-up self–tests are available on demand by power cycling the module.

On power-up or reset, the module performs the self-tests that are described in Table 14 below. All KATs must be completed successfully prior to any other use of cryptography by the Module. If one of the KATs fails, the module enters the error state. The module will output a detailed error message when FipsModeIndicator.errorCode() is called. The error state can only be cleared by reloading the module and calling FipsModeIndicator.isReady() again to confirm successful completion of the KATs.

Table 10 – Power Up Self-tests

Test Target	Description
Software Integrity	HMAC-SHA-256
AES	KATs: Encryption, Decryption Modes: CBC, ECB Key sizes: 128 bits
AES-FF1	KATs: Encryption, Decryption Key sizes: 128 bits
HMAC-SHA-256	KATs: Output verification SHA sizes: SHA-256
SHA	KATs: Output verification SHA sizes: SHA-256
KBKDF	KATs: Output verification

4.4 Physical Security Policy

Physical Security Mechanisms	**Recommended Frequency of Inspection/Test**	**Inspection/Test Guidance Details**
N/A	N/A	N/A

4.5 Operational Environment

The module operates in a modifiable operational environment under the FIPS 140-2 definitions. The module runs on a GPC running one of the operating systems specified in the approved operational environment list. Each approved operating system manages processes and threads in a logically separated manner. The Module’s user is considered the owner of the calling application that instantiates the Module within the process space of the Java Virtual Machine.

5. Mitigation of Other Attacks Policy

Other Attacks	**Mitigation Mechanism**	**Specific Limitations**
N/A	N/A	N/A

6. Security Rules and Guidance

6.1 Basic Enforcement

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 140-2 Level 1,2 and 3 module.

1. The module shall provide two distinct operator roles: User and Cryptographic Officer.
2. The module does not provide authentication.
3. The operator shall be capable of commanding the module to perform the power up self-tests by cycling power or resetting the module.

4. Power up self-tests do not require any operator action.
5. Data output shall be inhibited during key generation, self-tests, zeroization, and error states.
6. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
7. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
8. The module supports single user.
9. The module does not have any external input/output devices used for entry/output of data.
10. The module does not enter or output plaintext CSPs from the module’s physical boundary.
11. The module does not output intermediate key values.
12. The module does not allow concurrent operators.

6.2 Basic Guidance

The jar file representing the module needs to be installed in a JVM's classpath in a manner appropriate to its use in applications running on the JVM.

Functionality in the module is provided by the distinct classes that provide access to the FIPS approved and non-FIPS approved services provided by the module.

When the module is being used in FIPS approved-only mode, classes providing implementations of algorithms which are not FIPS approved, or allowed, are explicitly disabled.

7. EMI/EMC

The cryptographic module conforms to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B.