PreVeil Cryptographic Module

FIPS 140-2 Non-Proprietary Security Policy

Security Level 1 Validation

PreVeil, Inc.

Document Version 1.07

Module Version 2.0.0

January 2021

Table of Contents, Table of Figures, List of Tables

Table of Contents

Table of Figures

Table of Tables

1. Overview

This document is a non-proprietary FIPS 140-2 Security Policy for the PreVeil Cryptographic Module. This policy describes how the PreVeil Cryptographic Module (hereafter referred to as “fips-crypto” or “module”) meets the requirements of FIPS 140-2. This document also describes how to configure the module into the FIPS 140-2 Approved mode. This document was prepared by Leidos as part of a FIPS 140-2 Security Level 1 validation.

The Federal Information Processing Standards Publication 140-2 - Security Requirements for Cryptographic Modules (FIPS 140-2) details the United States Federal Government requirements for cryptographic modules. Detailed information about the FIPS 140-2 standard and validation program is available on the NIST (National Institute of Standards and Technology) website at https://csrc.nist.gov/projects/cryptographic-module-validation-program

2. Introduction

PreVeil Cryptographic module is a PreVeil code module that provides various cryptographic operations in a secure, uniform way to the other components in the PreVeil SaaS platform and client software that make up PreVeil's end-to-end encrypted messaging and file sharing service currently available for free individual and paid enterprise use. The PreVeil Cryptographic Module is being validated as a multi-chip standalone cryptographic module at FIPS 140-2 overall Security Level 1.

3. Cryptographic Module Specification

3.1. Security Level Summary

The security level claimed for each section of the FIPS 140-2 standard are as follows:

Section	Title	Level
1	Cryptographic Module Specification	1
2	Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A[1]
Overall		**1**

Figure 1 Security Level Summary

3.2. Cryptographic Boundary

The physical cryptographic boundary for the PreVeil Cryptographic Module is the edge (front, back, left,

[1]There are no special mechanisms built into or designed into the module to mitigate any specific attacks beyond those required by the FIPS 140-2 standard.

right, top, and bottom surfaces) of the physical enclosure for the physical appliance that the module is running on. The logical cryptographic boundary for the module is the fips-crypto library (libfips-crypto.dll) file itself.

3.3. Block Diagram

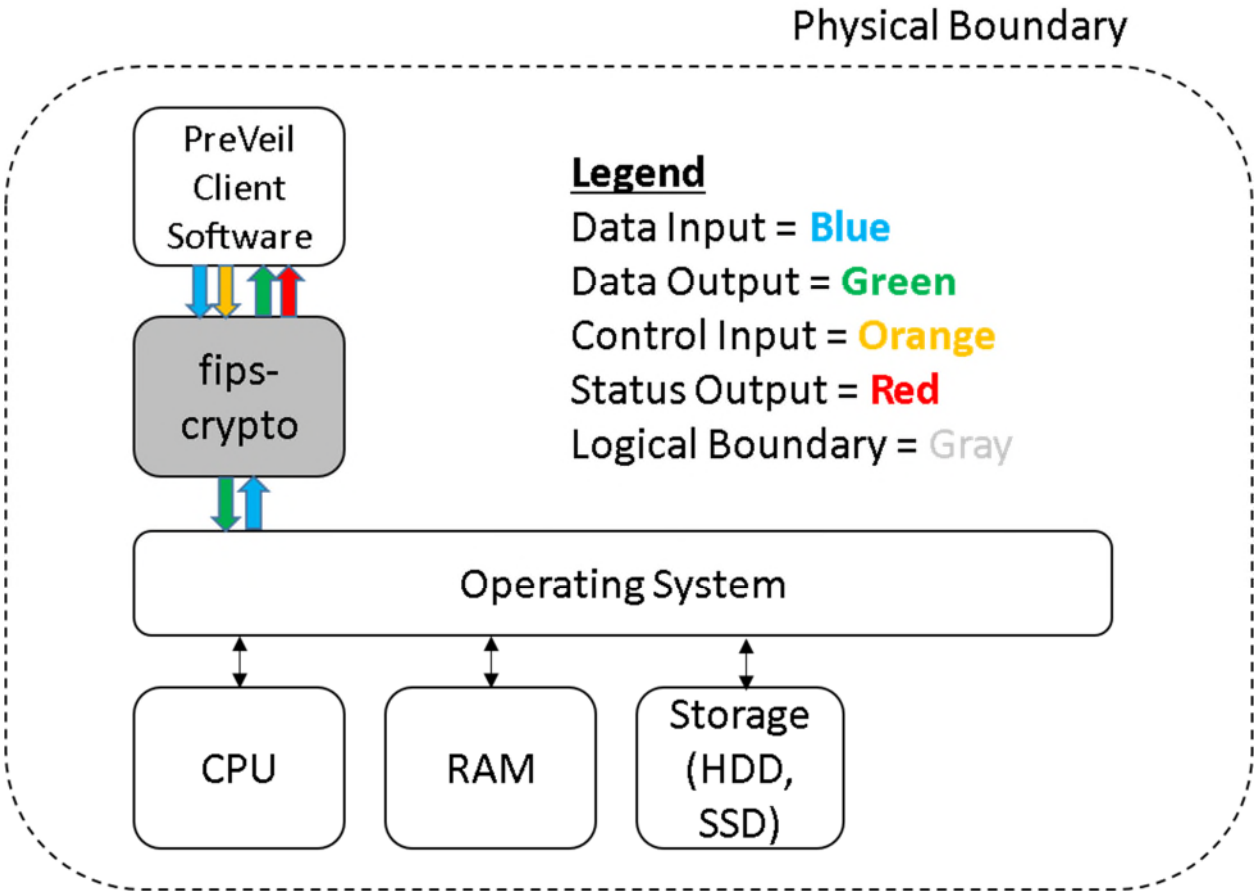

Figure 2 Block Diagram

3.4. Secure Initialization

The following steps must be followed to initialize the module into the FIPS Approved mode of operation. No special installation or start-up procedures need to be performed beyond just loading the module via a calling application and adhering to the steps below.

- The calling application of the module must use Approved algorithms *(see section Approved Algorithms*) in contexts which require security. No non-Approved algorithms shall be used in a context which requires security.
- Per the guidance in **FIPS 140-2 IG 1.23**, calling applications may use non-Approved algorithms in contexts where security is not required or being claimed.
- When utilizing the module, the following APIs are whitelisted and Approved for usage in the module's FIPS Approved mode of operation. Anything outside of this whitelist (below) of APIs is disallowed by policy. The full list of APIs supported by the module can be obtained from PreVeil's FIPS Crypto API specification:
    - **fips_crypto_free**
    - **fips_crypto_last_error**
    - **aes_encrypt_init**

- **aes_encrypt_update**
- **aes_encrypt_finalize**
- **aes_decrypt_init**
- **aes_decrypt_update**
- **aes_decrypt_finalize**
- **generate_ec_key**
- **ec_key_to_binary**
- **ec_key_from_binary**
- **ec_key_free**
- **ec_sign**
- **ec_verify**
- **box_encrypt**
- **box_decrypt**
- **box_derive_key**
- **hybrid_encrypt**
- **hybrid_decrypt**
- **verify_checksums**

- When making usage of the '**box_encrypt', 'box_decrypt', 'box_derive_key', 'hybrid_encrypt',** or **'hybrid_decrypt'** APIs, the 'kdf_type' argument must be set to a value of 'KDF_FIPS'. Anything outside of this whitelisted value is disallowed by policy.
- When making usage of the '**hybrid_encrypt**' and '**hybrid_decrypt**' APIs, the 'use_iv' argument must be set to a value of 'true'. Anything outside of this whitelisted value is disallowed by policy.

Failure to follow the above procedures will result in the module operating in a non-approved manner. When transitioning the module between the FIPS Approved mode *(i.e.: adherence to the steps outlined above*) and being utilized in a non-approved manner, the operator of the module must zeroize all Keys and CSPs. Similarly, in order to re-establish a FIPS Approved mode after utilizing the module in a non-approved manner, the operator shall once again zeroize all Keys and CSPs and adhere to the steps outlined above.

3.5. Approved Algorithms

The module supports the following approved algorithms for use in the approved mode. Although the module's cryptographic implementation supports more options than listed below, only those listed are usable by the module's APIs.

CAVP Cert	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli	Use
#C1714	AES[2]	FIPS 197 SP 800-38D	ECB[3], GCM[4]	256	Data Encryption / Decryption
Vendor Affirmed	CKG	SP 800-133 rev1	Section 5		Asymmetric Key Generation
#C1714	DRBG	SP 800-90A	CTR_DRBG w/ AES-256		Deterministic Random Bit Generation

[2] The module supports the use of AES-NI hardware acceleration if available.

[3] While the module supports AES-ECB, it is only used in the context that it is a prerequisite for GCM mode as well as the module's CTR_DRBG. The module does not provide a standalone ECB encryption/decryption service.

[4] The module supports AES-GCM encryption using an internally generated IV. The 96-bit IV is generated using the module's Approved DRBG (which is seeded from within the module's physical boundary with 256 bits of entropy and supports a security strength of 256 bits). This is compliant with FIPS 140-2 IG A.5 scenario 2.

#C1714	ECDSA	FIPS 186-4	-	P-256 (w/ SHA-256)	ECC Key Generation[5] Signature Generation Signature Verification
#C1714	HMAC	FIPS 198-1	HMAC-SHA-256	256	Integrity Check
Vendor Affirmed	KAS-SSC	SP 800-56A rev3		P-256[6]	Key Agreement
Vendor Affirmed	KDA	SP 800-56C rev1	One-step KDF using SHA-256	256	Key Derivation
#C1714	SHS	FIPS 180-4	SHA-256		Message Digest

Table 1 Approved Algorithms

3.6. Allowed Algorithms

The following algorithms are non-approved but allowed for use in the approved mode.

Algorithm	Caveat	Use
NDRNG	This implementation satisfies scenario 1(b) of IG 7.14. The module obtains a minimum of **256** bits of entropy before generating keys.	Seeding the DRBG

Table 2 Allowed Algorithms

3.7. Non-Approved Algorithms Table

The following algorithms are non-approved for use in the approved mode in any context requiring security. See section Secure Initialization for more detail on acceptable usage.

Algorithm	Caveat	Use
EdDSA (with Curve25519)	No security claimed	Signature Generation Signature Verification
X25519	No security claimed	Key Agreement

Table 3 Non-Approved Algorithms

4. Cryptographic Module Ports and Interfaces

4.1. Logical Interfaces

The module's interfaces can be categorized under the following FIPS 140-2 logical interfaces.

- Data Input
- Data Output
- Control Input

[5] The ECC keys used for EC Diffie-Hellman and ECDSA are generated according to FIPS 186-4.

[6] KAS-SSC with P-256 is used to produce AES-256 keys. Due to P-256 providing an equivalent security strength of only 128 bits (vs. the expected 256 for an AES-256 key), the following caveat applies to KAS-SSC per FIPS 140-2 IG 7.5: Key establishment methodology provides 128 bits of encryption strength

- Status Output

The following table provides a mapping of the module's interfaces to the FIPS 140-2 defined interface categories.

FIPS 140-2 Logical Interface(s)	PreVeil Cryptographic Module Interface
Data Input	API Input Parameters
Data Output	API Output Parameters
Control Input	Exported API Functions
Status Output	API Return Values Specific Exported API Functions
Power Input	N/A

Table 4 Logical Interfaces

5. Roles, Services, and Authentication

5.1. Roles

The PreVeil Cryptographic Module does not implement any form of authentication of users, as no authentication is required for a Level 1 module. The PreVeil Cryptographic Module defines a logical 'User' and 'Crypto Officer' role, both of which have access to the same set of services. The only differentiator between a User and Crypto Officer would be whether the operator (i.e.: calling application) is calling with elevated privileges (like 'Administrator' on Windows) or not.

5.2. Services

Listed below are the services for each of the module's roles that are approved for use in the FIPS approved mode. The access qualifiers that appear in the Key/CSP Access column are described as follows:

- **Write**: The Key/CSP is either written for the first time (like by being input into the API), or is otherwise overwritten by this service
- **Execute**: The Key/CSP is utilized by this service in a cryptographic operation
- **Agree**: The Key/CSP is established by this service using an Approved key agreement scheme
- **Derive**: The Key/CSP is derived by this service using an Approved key derivation algorithm
- **Generate**: The Key/CSP is generated by this service using an Approved key generation algorithm
- **Delete**: The Key/CSP is zeroized by this service

5.2.1. User and Crypto-Officer Services

Name	Associated API(s)	Description	Inputs	Outputs	Key/CSP Access
AES Key Establishment	box_derive_key	Produce AES-GCM key	Key Pairs KDF Type	Derived Key	• ECDH Public Key (write, execute) • ECDH Private

					Key (write, execute) • Shared Secret (agree, execute) • AES Key (derive) • DRBG CSPs (write, execute)
AES Encryption	aes_encrypt_init aes_encrypt_update aes_encrypt_finalize box_encrypt hybrid_encrypt	Perform AES-GCM encryption based on a derived **or** entered key	Key Pairs KDF Type Key Plaintext	Ciphertext	• All Keys/CSPs from AES Key Establishment row if 'box' or 'hybrid' APIs are used • AES Key (write, execute) • DRBG CSPs (write, execute)
AES Decryption	aes_decrypt_init aes_decrypt_update aes_decrypt_finalize box_decrypt hybrid_decrypt	Perform AES-GCM decryption based on a derived **or** entered key	Key Pairs KDF Type Key IV Ciphertext	Plaintext	• All Keys/CSPs from AES Key Establishment row if 'box' or 'hybrid' APIs are used • AES Key (write, execute)
EC Key Generation	generate_ec_key ec_key_to_binary	Generate Elliptic Curve Key Pairs for ECDSA / ECDH	Key Type Key Usage	Key Pair	• ECDSA Public Key (generate) • ECDSA Private Key (generate) • ECDH Public Key (generate) • ECDH Private Key (generate) • DRBG CSPs (write, execute)

Digital Signature Generation	ec_key_from_binary ec_sign	Generate ECDSA digital signature	Key Pair Message	Signature	• ECDSA Private Key (write, execute) • DRBG CSPs (write, execute)
Digital Signature Verification	ec_key_from_binary ec_verify	Verify ECDSA digital signature	Key Pair Message Signature	True or False	• ECDSA Public Key (write, execute)
Zeroization	aes_encrypt_finalize aes_decrypt_finalize ec_key_free fips_crypto_free	Zeroize key material	Memory Reference	None	• All Keys and CSPs (delete)
Show Status	fips_crypto_last_error	Output status information	None	Status String	• None
Self-Tests	DllMain (Default Entry Point)	Load the module and execute the power-on self-tests (automatic)	None	Status String	• Integrity Test Key (execute)

Table 5 User and Crypto-Officer Services

5.2.2. Non-Approved Services

The following services are non-approved for use in the FIPS approved mode, but are otherwise available to the User and Crypto-Officer roles.

Name	Description
AES Key Establishment with Non-Approved KDF	The module supports multiple methods for key derivation for AES-GCM keys. Per FIPS 140-2 Annex D, only the module's SP 800-56C Rev. 1 KDF is Approved. Using any other KDF type beyond what is designated in Secure Initialization is Non-Approved and is disallowed by policy.
AES Encryption with External IV	The module supports two methods for IVs for AES-GCM encryption; internal generation and external input. Per FIPS 140-2 IG A.5, the external input of IVs is not allowed for usage in the FIPS Approved mode.
Non-Whitelisted Exported APIs	This Security Policy (section Secure Initialization) provides a whitelist of the exported APIs that can be utilized in the FIPS approved mode. The usage of any APIs outside of this white list is considered to be Non-Approved and is disallowed by policy.

Table 6 Non-approved Services

5.3. Authentication

The module does not support any authentication methods as it is a Level 1 module. Roles are logically assumed based on the privilege level of the calling application as determined by the operating system.

6. Physical Security

The module is a software module whose host must run on a production grade platform (e.g. commercially made server or general purpose computer).

7. Operational Environment

The module is operating in a modifiable operational environment. For this FIPS 140-2 certification effort, the module was tested on the following platforms:

- **Microsoft Windows 10 (32-bit on x86-64) running on a Dell XPS 8700 with an Intel Core i7 with PAA (AES-NI)**
- **Microsoft Windows 10 (64-bit on x86-64) running on a Dell XPS 8700 with an Intel Core i7 with PAA (AES-NI)**
- **Microsoft Windows 10 (32-bit on x86-64) running on a Dell XPS 8700 with an Intel Core i7 without PAA (AES-NI)**
- **Microsoft Windows 10 (64-bit on x86-64) running on a Dell XPS 8700 with an Intel Core i7 without PAA (AES-NI)**

8. Cryptographic Key Management

Key/CSP Name	Key/CSP Type	Key/CSP Size	Generation/ Input[7]	Output	Storage	Zeroization	Use[8]
AES Key	AES-GCM	256 bits	Input via the API in plaintext; Derived from Shared Secret	N/A	N/A	Zeroization Service	AES Encryption and AES Decryption
ECDH Public Key	P-256	256 bits	Generated; Input via the API in plaintext	Output when Generated	N/A	Zeroization Service	Producing Shared Secret via Key Agreement

[7]For all keys marked as "generated", the resulting symmetric key or the generated seed to be used in the asymmetric key generation is an unmodified output from the DRBG unless otherwise noted.
[8]Keys/CSPs established in FIPS mode shall not be used in a non-Approved mode/service and vice-versa.

ECDH Private Key	P-256	256 bits	Generated; Input via the API in plaintext	Output when Generated	N/A	Zeroization Service	Producing Shared Secret via Key Agreement
ECDSA Public Key	P-256	256 bits	Generated; Input via the API in plaintext	Output when Generated	N/A	Zeroization Service	Verifying Digital Signatures
ECDSA Private Key	P-256	256 bits	Generated; Input via the API in plaintext	Output when Generated	N/A	Zeroization Service	Generating Digital Signatures
Shared Secret	Shared Secret	256 bits	Established via Key Agreement	N/A	N/A	Zeroization Service	Deriving AES Keys
DRBG CSPs	Entropy Input, Seed, V and Key	Entropy Input (256 bits), Seed (384 bits), V (128 bits) and Key (256 bits)	Entropy Input is Generated via NDRNG	N/A	N/A	Zeroization Service	Generating Random Numbers
Integrity Test Key	HMAC-SHA-256	256 bits	Hard-Coded	N/A	Hard-Coded	N/A (Not Required per FIPS 140-2 IG 7.4)	Integrity Check

Table 7 Cryptographic Keys and CSPs

9. EMI / EMC

The tested platform conformed to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (i.e., for business use).

10. Self-Tests

Output via the Data Output interface is inhibited during the performance of self-tests. The module enters the error state upon any self-test failure. The following self-tests are executed automatically without any need for input or actions from the user.

10.1. Power-on Self-Tests

The following self-tests are automatically executed upon loading the module (via the module's default entry point), and can be repeated on demand by reloading the module:

- Integrity Test (HMAC-SHA-256 with 256-bit Key)
- SHA-256 Known Answer Test
- HMAC-SHA-256 Known Answer Test
- AES-256 ECB encrypt / decrypt Known Answer Test
- AES-256 GCM encrypt / decrypt Known Answer Test
- ECDSA sign / verify Pairwise Consistency Test (P-256 with SHA-256)
- CTR_DRBG w/ AES-256 Known Answer Tests (Instantiate, Reseed, Generate)
- Primitive "Z" Computation Known Answer Test for Elliptic-Curve Diffie-Hellman (P-256)

10.2. Conditional Self-Tests

- Health Tests (Instantiate, Reseed, Generate) on the SP800-90A CTR_DRBG w/ AES-256
- Repetition Count Test (RCT) on the NDRNG
- Adaptive Proportion Test (APT) on the NDRNG
- ECDSA Pair-wise Consistency Test
- Elliptic-Curve Diffie-Hellman Pair-wise Conditional Test
- Conditional Tests for Assurances (as specified in SP800-56A Sections 5.5.2, 5.6.2 and 5.6.3)

A. Appendices

Table of Acronyms:

Acronym	Definition
AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard – New Instructions
API	Application Programming Interface
APT	Adaptive Proportion Test
CAVP	Cryptographic Algorithm Validation Program
CKG	Cryptographic Key Generation
CPU	Central Processing Unit
CRNGT	Continuous Random Number Generator Test
CSP	Critical Security Parameter
CVL	Component Validation List
DRBG	Deterministic Random Bit Generator
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
EMI	Electromagnetic Interference
EMC	Electromagnetic Compatibility
FIPS	Federal Information Processing Standard
GCM	Galois Counter Mode
HDD	Hard Disk Drive
HMAC	Keyed-Hash Message Authentication Code
IG	Implementation Guidance
KAS	Key Agreement Scheme
KAS-SSC	Key Agreement Scheme – Shared Secret Computation
KDA	Key Derivation Algorithm
KDF	Key Derivation Function
NDRNG	Non-Deterministic Random Number Generator
RAM	Random Access Memory
RCT	Repetition Count Test
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SSD	Solid State Drive