Google Inc.

BoringCrypto

FIPS 140-2 Security Policy

Software version: 24e5886c0edfc409c8083d10f9f1120111efd6f5

Date: July 18th, 2017



Prepared by:

Acumen Security

18504 Office Park Dr.

Montgomery Village, MD 20886

[www.acumensecurity.net](http://www.acumensecurity.net)

# Introduction

Federal Information Processing Standards Publication 140-2 — Security Requirements for Cryptographic Modules specifies requirements for cryptographic modules to be deployed in a Sensitive but Unclassified environment. The National Institute of Standards and Technology (NIST) and Communications Security Establishment Canada (CSE) Cryptographic Module Validation Program (CMVP) run the FIPS 140 program. The NVLAP accredits independent testing labs to perform FIPS 140 testing; the CMVP validates modules meeting FIPS 140 validation. Validated is the term given to a module that is documented and tested against the FIPS 140 criteria.

More information is available on the CMVP website at: <http://csrc.nist.gov/groups/STM/cmvp/index.html>.

# About this Document

This non-proprietary Cryptographic Module Security Policy for BoringCrypto from Google Inc. provides an overview of the product and a high-level description of how it meets the overall Level 1 security requirements of FIPS 140-2.

BoringCrypto module may also be referred to as the “module” in this document.

# Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design, and manufacturing. Google Inc. shall have no liability for any error or damages of any kind resulting from the use of this document.

The software referenced in this document is licensed to you under the terms and conditions accompanying the software or as otherwise agreed between you or the company that you are representing.

# Notices

This document may be freely reproduced and distributed in its entirety without modification.

Table of Contents

[Introduction 2](#_Toc487029666)

[About this Document 2](#_Toc487029667)

[Disclaimer 2](#_Toc487029668)

[Notices 2](#_Toc487029669)

[1. Introduction 6](#_Toc487029670)

[2. FIPS 140-2 Security Levels 7](#_Toc487029671)

[3. Cryptographic Module Specification 8](#_Toc487029672)

[3.1 Cryptographic Boundary 8](#_Toc487029673)

[4. Modes of Operation 9](#_Toc487029674)

[5. Cryptographic Module Ports and Interfaces 9](#_Toc487029675)

[6. Roles, Authentication and Services 10](#_Toc487029676)

[7. Physical Security 12](#_Toc487029677)

[8. Operational Environment 12](#_Toc487029678)

[9. Cryptographic Algorithms & Key Management 13](#_Toc487029679)

[9.1 Approved Cryptographic Algorithms 13](#_Toc487029680)

[9.2 Allowed Cryptographic Algorithms 13](#_Toc487029681)

[9.3 Non-Approved Cryptographic Algorithms 13](#_Toc487029682)

[9.4 Cryptographic Key Management 14](#_Toc487029683)

[9.5 Public Keys 14](#_Toc487029684)

[9.6 Key Generation 15](#_Toc487029685)

[9.7 Key Storage 15](#_Toc487029686)

[9.8 Key Zeroization 15](#_Toc487029687)

[10. Self-tests 16](#_Toc487029688)

[10.1 Power-On Self-Tests 16](#_Toc487029689)

[10.2 Conditional Self-Tests 16](#_Toc487029690)

[11. Mitigation of other Attacks 17](#_Toc487029691)

[12. Guidance and Secure Operation 18](#_Toc487029692)

[12.1 Installation Instructions 18](#_Toc487029693)

[12.2 Secure Operation 19](#_Toc487029694)

[12.2.1 Initialization 19](#_Toc487029695)

[12.2.2 Usage of AES-GCM 19](#_Toc487029696)

[12.2.3 TLS Operations 19](#_Toc487029697)

[12.2.4 Usage of Triple-DES 19](#_Toc487029698)

[12.2.5 RSA and ECDSA Keys 19](#_Toc487029699)

[13. References and Standards 20](#_Toc487029700)

[14. Acronyms and Definitions 20](#_Toc487029701)

List of Tables

[Table 1 – Tested Operational Environments 6](#_Toc487029702)

[Table 2 – Validation Level by FIPS 140-2 Section 7](#_Toc487029703)

[Table 3 – Ports and Interfaces 9](#_Toc487029704)

[Table 4 – Approved Services, Roles and Access Rights 10](#_Toc487029705)

[Table 5 – non-Approved or non-security relevant services 10](#_Toc487029706)

[Table 6 - Non-Security Relevant Services 11](#_Toc487029707)

[Table 7 – Approved Algorithms and CAVP Certificates 13](#_Toc487029708)

[Table 8 – Allowed Algorithms 13](#_Toc487029709)

[Table 9 – Non-Approved Algorithms 14](#_Toc487029710)

[Table 10 – Keys and CSPs supported 14](#_Toc487029711)

[Table 11 – Public keys supported 14](#_Toc487029712)

[Table 12 – Power-on Self-tests 16](#_Toc487029713)

[Table 13 – Conditional Self-tests 16](#_Toc487029714)

[Table 14 – References and Standards 20](#_Toc487029715)

[Table 15 – Acronyms and Definitions 21](#_Toc487029716)

List of Figures

[Figure 1 – Logical Boundary 8](#_Toc487029717)

# Introduction

Google Inc. BoringCrypto module (hereafter referred to as the “module”) is an open-source, general-purpose cryptographic library which provides FIPS 140-2 approved cryptographic algorithms to serve BoringSSL and other user-space applications. The validated version of the library is *24e5886c0edfc409c8083d10f9f1120111efd6f5*. For the purposes of the FIPS 140-2 validation, its embodiment type is defined as multi-chip standalone.

The cryptographic module was tested on the following operational environments on the general-purpose computer (GPC) platforms detailed below:

|  |  |  |  |
| --- | --- | --- | --- |
| # | Operational Environment | Processor Family | Compiler |
| 1 | Ubuntu Linux 14.04 LTS | Intel Xeon E5 (without PAA) | Clang (4.0.0) |
| 2 | Ubuntu Linux 16.04 | Intel Xeon E5 | Clang (4.0.0) |
| 3 | Ubuntu Linux 15.04 | POWER8 (without PAA) | Clang (4.0.0) |
| 4 | Ubuntu Linux 17.04 | POWER8 | Clang (4.0.0) |
| 4 | Ubuntu Linux 17.04 | POWER9 | Clang (4.0.0) |

Table – Tested Operational Environments

The cryptographic module is also supported on the following operating environments for which operational testing and algorithm testing was not performed:

* Linux 3.X executing on x86\_64 architecture;
* Linux 4.X executing on x86\_64 architecture;
* Linux 3.X executing on POWER8 architecture;
* Linux 4.X executing on POWER8 architecture;
* Linux 3.X executing on POWER9 architecture; and
* Linux 4.X executing on POWER9 architecture.

As per FIPS 140-2 Implementation Guidance G.5, compliance is maintained for other versions of the respective operational environments where the module binary is unchanged. No claim can be made as to the correct operation of the module or the security strengths of the generated keys if any source code is changed and the module binary is reconstructed.

The GPC(s) used during testing met Federal Communications Commission (FCC) FCC Electromagnetic Interference (EMI) and Electromagnetic Compatibility (EMC) requirements for business use as defined by 47 Code of Federal Regulations, Part 15, Subpart B. FIPS 140-2 validation compliance is maintained when the module is operated on other versions of the GPOS running in single user mode, assuming that the requirements outlined in NIST IG G.5 are met.

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

# FIPS 140-2 Security Levels

The following table lists the level of validation for each area in FIPS 140-2:

| **FIPS 140-2 Section Title** | **Validation Level** |
| --- | --- |
| Cryptographic Module Specification | 1 |
| Cryptographic Module Ports and Interfaces | 1 |
| Roles, Services, and Authentication | 1 |
| Finite State Model | 1 |
| Physical Security | 1 |
| Operational Environment | N/A |
| Cryptographic Key Management | 1 |
| Electromagnetic Interference / Electromagnetic Compatibility | 1 |
| Self-Tests | 1 |
| Design Assurance | 1 |
| Mitigation of Other Attacks | N/A |
| Overall Level | 1 |

Table – Validation Level by FIPS 140-2 Section

# Cryptographic Module Specification

## Cryptographic Boundary

The module is a software library providing a C-language application program interface (API) for use by other processes that require cryptographic functionality. All operations of the module occur via calls from host applications and their respective internal daemons/processes. As such there are no untrusted services calling the services of the module.

The physical cryptographic boundary is the general-purpose computer on which the module is installed. The logical cryptographic boundary of the BoringCrypto module is a single object file named bcm.o which is statically linked to BoringSSL. The module performs no communications other than with the calling application (the process that invokes the module services) and the host operating system.

Figure 1 shows the logical relationship of the cryptographic module to the other software and hardware components of the computer.



Figure – Logical Boundary

# Modes of Operation

The module supports two modes of operation: Approved and Non-approved. The module will be in FIPS-approved mode when all power up self-tests have completed successfully and only Approved algorithms are invoked. See Table 7 below for a list of the supported Approved algorithms and Table 8 for allowed algorithms. The non-Approved mode is entered when a non-Approved algorithm is invoked. See Table 9 for a list of non-Approved algorithms.

# Cryptographic Module Ports and Interfaces

The Data Input interface consists of the input parameters of the API functions. The Data Output interface consists of the output parameters of the API functions. The Control Input interface consists of the actual API input parameters. The Status Output interface includes the return values of the API functions.

| **FIPS Interface** | **Physical Ports** | **Logical Interfaces** |
| --- | --- | --- |
| Data Input | Physical ports of the tested platforms | API input parameters |
| Data Output | Physical ports of the tested platforms | API output parameters and return values |
| Control Input | Physical ports of the tested platforms | API input parameters |
| Status Output | Physical ports of the tested platforms | API return values |
| Power Input | Physical ports of the tested platforms | N/A |

Table – Ports and Interfaces

As a software module, control of the physical ports is outside module scope. However, when the module is performing self-tests, or is in an error state, all output on the module’s logical data output interfaces is inhibited.

# Roles, Authentication and Services

The cryptographic module implements both User and Crypto Officer (CO) roles. The module does not support user authentication. The User and CO roles are implicitly assumed by the entity accessing services implemented by the module. A user is considered the owner of the thread that instantiates the module and, therefore, only one concurrent user is allowed.

The Approved services supported by the module and access rights within services accessible over the module’s public interface are listed in the table below.

| **Service** | **Approved security functions** | **Keys and/or CSPs** | **Roles** | **Access rights to keys and/or CSPs** |
| --- | --- | --- | --- | --- |
| Module Initialization | N/A | N/A | CO | N/A |
| Symmetric encryption/decryption | AES, TDES | AES, TDES symmetric keys | User, CO | Execute |
| Keyed hashing | HMAC-SHA | HMAC key | User, CO | Execute |
| Hashing | SHS | None | User, CO | N/A |
| Random Bit Generation | CTR\_DRBG | DRBG seed, internal state V and Key values | User, CO | Write/Execute |
| Signature generation/ verification | CTR\_DRBG  RSA  ECDSA | RSA, ECDSA private key | User, CO | Write/Execute |
| Key Transport | RSA | RSA private key | User, CO | Write/Execute |
| Key Generation | CTR\_DRBG  RSA  ECDSA | RSA, ECDSA private key | User, CO | Write/Execute |
| On-Demand  Self-test | None | None | User, CO | Execute |
| Zeroization | None | All keys | User, CO | Write/Execute |
| Show status | None | None | User, CO | N/A |

Table – Approved Services, Roles and Access Rights

The module provides the following non-Approved services which utilize algorithms listed in Table 9:

| **Service** | **Non-Approved Functions** | **Roles** | **Keys and/or CSPs** |
| --- | --- | --- | --- |
| Symmetric encryption/decryption | AES, DES, TDES | User, CO | N/A |
| Hashing | MD4, MD5, POLYVAL | User, CO | N/A |
| Signature generation/ verification | RSA  ECDSA | User, CO | N/A |
| Key Transport | RSA | User, CO | N/A |
| Key Generation | RSA  ECDSA | User, CO | N/A |

Table – non-Approved or non-security relevant services

The module also provides the following non-Approved or non-security relevant services over a non-public interface:

| **Service** | **Approved Security**  **Functions** | **Roles** | **Access rights to keys**  **and/or CSPs** |
| --- | --- | --- | --- |
| Large integer operations | None | User, CO | N/A |
| Disable automatic generation of CTR\_DRBG "additional\_input" parameter" | CTR\_DRBG | User, CO | N/A |
| Wegman-Carter hashing with POLYVAL | None | User, CO | N/A |

Table - Non-Security Relevant Services

# Physical Security

The cryptographic module is comprised of software only and thus does not claim any physical security.

# Operational Environment

The cryptographic module operates under Ubuntu Linux 14.04 LTS, 15.04, 16.04 and 17.04. The module runs on a GPC running one of the operating systems specified in Table 1. 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.

# Cryptographic Algorithms & Key Management

## Approved Cryptographic Algorithms

The module implements the following FIPS 140-2 Approved algorithms:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CAVP Cert #** | **Algorithm** | **Standard** | **Mode/Method** | **Use** |
| 4558 | AES | SP 800-38A  FIPS 197  SP 800-38F  SP 800-38D | CBC, ECB, CTR, GCM, KW | Encryption, Decryption, Key Wrapping, Key Unwrapping, Decryption, Authentication |
| 2428 | Triple-DES | SP 800-20 | TCBC, TECB | Encryption, Decryption |
| 1112  1240 (CVL) | ECDSA | FIPS 186-4 | Signature Generation Component, Key Pair Generation, Signature Generation, Signature Verification, Public Key Validation | Digital Signature Services |
| 3011 | HMAC | FIPS 198-1 | HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 | Generation, Authentication |
| 3736 | SHA | FIPS 180-4 | SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 | Digital Signature Generation, Digital Signature Verification, non-Digital Signature Applications |
| 1507 | DRBG | SP 800-90Arev1 | CTR\_DRBG | Random Bit Generation |
| 2485 | RSA | FIPS 186-4 | Key Generation, Signature Generation, Signature Verification | Digital Signature Services |

Table – Approved Algorithms and CAVP Certificates

## Allowed Cryptographic Algorithms

The module supports the following non-FIPS 140-2 Approved but allowed algorithms that may be used in the Approved mode of operation.

|  |  |
| --- | --- |
| Algorithm | Use |
| RSA Key Transport | RSA key wrapping (key establishment methodology provides between 112 and 256 bits of encryption strength) |
| NDRNG | Used only to seed the Approved DRBG |

Table – Allowed Algorithms

## Non-Approved Cryptographic Algorithms

The module employs the methods listed in Table 9, which are not allowed for use in a FIPS-Approved mode. Their use will result in the module operating in a non-Approved mode.

|  |  |
| --- | --- |
| MD5, MD4 | DES |
| AES-GCM (non-compliant) | AES (non-compliant) |
| ECDSA (non-compliant) | RSA (non-compliant) |
| POLYVAL | Triple-DES (non-compliant) |

Table – Non-Approved Algorithms

## Cryptographic Key Management

The table below provides a complete list of Private Keys and CSPs used by the module:

| Key/CSP Name | Key Description | Generated/ Input | Output |
| --- | --- | --- | --- |
| AES Key | AES (128/192/256) encrypt / decrypt key | Input via API in plaintext | Output via API in plaintext |
| AES-GCM Key | AES (128/192/256) encrypt / decrypt / generate / verify key | Input via API in plaintext | Output via API in plaintext |
| AES Wrapping Key | AES (128/192/256) key wrapping key | Input via API in plaintext | Output via API in plaintext |
| Triple-DES Key | Triple-DES (3-Key) encrypt / decrypt key | Input via API in plaintext | Output via API in plaintext |
| ECDSA Signing Key | ECDSA (P-224/P-256/P-384/P-521) signature generation key | Internally Generated or input via API in plaintext | Output via API in plaintext |
| HMAC Key | Keyed hash key (160/224/256/384/512) | Input via API in plaintext | Output via API in plaintext |
| RSA Key (Key Transport) | RSA (2048 to 16384 bits) key decryption (private key transport) key | Internally Generated or input via API in plaintext | Output via API in plaintext |
| RSA Signature Generation Key | RSA (2048 to 16384 bits) signature generation key | Internally Generated or input via API in plaintext | Output via API in plaintext |
| CTR\_DRBG V (Seed) | 128 bits | Internally Generated | Does not exit the module |
| CTR\_DRBG Key | 256 bits | Internally Generated | Does not exit the module |
| CTR\_DRBG Entropy Input | 384 bits | Input via API in plaintext | Does not exit the module |

Table – Keys and CSPs supported

## Public Keys

The table below provides a complete list of the Public keys used by the module:

| Public Key Name | Key Description |
| --- | --- |
| ECDSA Verification Key | ECDSA (P-224/P-256/P-384/P-521) signature verification key |
| RSA Key (Key Transport) | RSA (2048 to 16384 bits) key encryption (public key transport) key |
| RSA Signature Verification Key | RSA (1024 to 16384 bits) signature verification public key |

Table – Public keys supported

## Key Generation

The module supports generation of ECDSA and RSA key pairs as specified in Section 5 of NIST SP 800-133. The module employs a NIST SP800-90A random number generator for creation of the seed for asymmetric key generation. The module requests a minimum number of 128 bits of entropy from its Operational Environment per each call.

The output data path is provided by the data interfaces and is logically disconnected from processes performing key generation or zeroization. No key information will be output through the data output interface when the module zeroizes keys.

## Key Storage

The cryptographic module does not perform persistent storage of keys. Keys and CSPs are passed to the module by the calling application. The keys and CSPs are stored in memory in plaintext. Keys and CSPs residing in internally allocated data structures (during the lifetime of an API call) can only be accessed using the module defined API. The operating system protects memory and process space from unauthorized access.

## Key Zeroization

The module is passed keys as part of a function call from a calling application and does not store keys persistently. The calling application is responsible for parameters passed in and out of the module. The Operating System and the calling application are responsible to clean up temporary or ephemeral keys.

# Self-tests

FIPS 140-2 requires the module to perform self-tests to ensure the integrity of the module and the correctness of the cryptographic functionality at start up. Some functions require conditional tests during normal operation of the module. The supported tests are listed and described in this section.

## Power-On Self-Tests

Power-on self-tests are run upon the initialization of the module and do not require operator intervention to run. If any of the tests fail, the module will not initialize. The module will enter an error state and no services can be accessed.

The module implements the following power-on self-tests:

| Type | Test |
| --- | --- |
| Integrity Test | HMAC-SHA-512 |
| Known Answer Test | AES KAT (encryption and decryption. Key size: 128-bits) |
| AES-GCM KAT (encryption and decryption. Key size: 128-bits) |
| Triple-DES KAT (encryption and decryption. Key size: 168-bits) |
| ECDSA KAT (signature generation/signature verification. Curve: P-256) |
| HMAC KAT (HMAC-SHA-1, HMAC-SHA-512) |
| SP 800-90A CTR\_DRBG KAT (Key size: 256-bits) |
| RSA KAT (signature generation/signature verification and encryption/decryption. Key size: 2048-bit) |
| SHA KAT (SHA-1, SHA-256, SHA-512) |

Table – Power-on Self-tests

Each module performs all power-on self-tests automatically when the module is initialized. All power-on self-tests must be passed before a User/Crypto Officer can perform services. The Power-on self-tests can be run on demand by power-cycling the host platform.

## Conditional Self-Tests

Conditional self-tests are run during operation of the module. If any of these tests fail, the module will enter an error state, where no services can be accessed by the operators. The module can be re-initialized to clear the error and resume FIPS mode of operation. Each module performs the following conditional self-tests:

| Type | Test |
| --- | --- |
| Pair-wise Consistency Test | ECDSA Key Pair generation  RSA Key Pair generation |
| CRNGT | Performed on NDRNG per IG 9.8 |
| DRBG Health Tests | Performed on DRBG, per SP 800‐90A Section 11.3. Required per IG C.1. |

Table – Conditional Self-tests

Pairwise consistency tests are performed for both possible modes of use, e.g. Sign/Verify and Encrypt/Decrypt.

# Mitigation of other Attacks

The module is not designed to mitigate against attacks which are outside of the scope of FIPS 140-2.

# Guidance and Secure Operation

## Installation Instructions

The following steps shall be performed to build, compile and statically link the BoringCrypto module to BoringSSL on the tested Operational Environments. The cryptographic module

The below tools are required in order to build and compile the module:

* Clang compiler version 4.0.0 (http://releases.llvm.org/download.html)
* Go programming language version 1.8 (https://golang.org/dl/)
* Ninja build system version 1.7.2 (https://github.com/ninja-build/ninja/releases)

Once the above tools have been obtained, issue the following command to create a CMake toolchain file to specify the use of Clang:

* printf "set(CMAKE\_C\_COMPILER \"clang\")\nset(CMAKE\_CXX\_COMPILER \"clang++\")\n" >> ${HOME}/toolchain

The FIPS 140-2 validated release of the module can be obtained by downloading the tarball containing the source code at the following location:

<https://commondatastorage.googleapis.com/chromium-boringssl-docs/fips/boringssl-24e5886c0edfc409c8083d10f9f1120111efd6f5.tar.xz>

or by issuing the following command:

* wget https://commondatastorage.googleapis.com/chromium-boringssl-docs/fips/boringssl-24e5886c0edfc409c8083d10f9f1120111efd6f5.tar.xz

The set of files specified in the archive constitutes the complete set of source files of the validated module. There shall be no additions, deletions, or alterations of this set as used during module build.

The downloaded tarball file can be verified using the below SHA-256 digest value:

15a65d676eeae27618e231183a1ce9804fc9c91bcc3abf5f6ca35216c02bf4da

By issuing the following command:

* sha256sum boringssl-24e5886c0edfc409c8083d10f9f1120111efd6f5.tar.xz

After the tarball has been extracted, the following commands will compile the module:

1. cd boringssl
2. mkdir build && cd build && cmake -GNinja -DCMAKE\_TOOLCHAIN\_FILE=${HOME}/toolchain -DFIPS=1 -DCMAKE\_BUILD\_TYPE=Release ..
3. ninja
4. ninja run\_tests

Upon completion of the build process. The module’s status can be verified by issuing:

* ./tool/bssl isfips

The module will print “1” if it is in a FIPS 140-2 validated mode of operation.

## Secure Operation

### Initialization

The cryptographic module is initialized by loading the module before any cryptographic functionality is available. In User Space the operating system is responsible for the initialization process and loading of the library. The module is designed with a default entry point (DEP) which ensures that the power-up tests are initiated automatically when the module is loaded.

### Usage of AES OFB, CFB and CFB8

In approved mode, users of the module must not utilize AES OFB, CFB and CFB8.

### Usage of AES-GCM

In approved mode, users of the module must not utilize GCM with an externally generated IV unless the source of the IV is also FIPS approved for GCM IV generation.

The module’s implementation of AES-GCM is used together with an application that executes outside of the module’s cryptographic boundary. The application negotiates the protocol session’s keys and the 32-bit nonce value of the IV. The nonce field and counter portions conform to the requirements in Provision 3 of IG A.5.

Per IG A.5, in the event module power is lost and restored the consuming application must ensure that any of its AES-GCM keys used for encryption or decryption are re-distributed.

### TLS Operations

The module does not implement the TLS protocol. It implements the cryptographic operations which can be used to implement the TLS protocol.

### Usage of Triple-DES

It is the calling application's responsibility to make sure that the three keys k1, k2 and k3 are independent. Two-key triple-DES usage will bring the module into the non-Approved mode of operation implicitly.

Per IG A.13, the module shall have a limit of either 2^32 or 2^28 encryptions with the same Triple-DES key. The calling application is responsible for ensuring the module’s compliance with this requirement.

### RSA and ECDSA Keys

The module allows the use of 1024 bits RSA keys for legacy purposes including signature generation, which is disallowed to be used in FIPS Approved mode as per NIST SP800-131A. Therefore, the cryptographic operations with the non-approved key sizes will result in the module operating in non-Approved mode implicitly.

Approved algorithms shall not use the keys generated by the module’s non-Approved key generation methods.

# References and Standards

The following Standards are referred to in this Security Policy.

| **Abbreviation** | **Full Specification Name** |
| --- | --- |
| FIPS 140-2 | Security Requirements for Cryptographic modules, May 25 2001 |
| FIPS 180-4 | Secure Hash Standard (SHS) |
| FIPS 186-4 | Digital Signature Standard (DSS) |
| FIPS 197 | Advanced Encryption Standard |
| FIPS 198-1 | The Keyed-Hash Message Authentication Code (HMAC) |
| FIPS 202 | SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions |
| IG | Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program, January 11, 2016. |
| SP 800-20 | Modes of Operation Validation System for Triple Data Encryption Algorithm (TMOVS) |
| SP 800-38A | Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode |
| SP 800-38D | Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC |
| SP 800-38F | Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping |
| SP 800-90A | Recommendation for Random Number Generation Using Deterministic Random Bit Generators |

Table – References and Standards

# Acronyms and Definitions

| **Acronym** | **Definition** |
| --- | --- |
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| CBC | Cipher-Block Chaining |
| CMAC | Cipher-based Message Authentication Code |
| CMVP | Crypto Module Validation Program |
| CO | Cryptographic Officer |
| CPU | Central Processing Unit |
| CSP | Critical Security Parameter |
| CTR | Counter-mode |
| CVL | Component Validation List |
| DES | Data Encryption Standard |
| DRAM | Dynamic Random Access Memory |
| DRBG | Deterministic Random Bit Generator |
| EC | Elliptic Curve |
| ECB | Electronic Code Book |
| ECDSA | Elliptic Curve Digital Signature Authority |
| FIPS | Federal Information Processing Standards |
| GCM | Galois/Counter Mode |
| GPC | General Purpose Computer |
| HMAC | key-Hashed Message Authentication Code |
| IG | Implementation Guidance |
| IV | Initialization Vector |
| KAT | Known Answer Test |
| MAC | Message Authentication Code |
| MD5 | Message Digest algorithm MD5 |
| N/A | Non Applicable |
| NDRNG | Non Deterministic Random Number Generator |
| OS | Operating System |
| RSA | Rivest Shamir Adleman |
| SHA | Secure Hash Algorithm |
| TDES | Triple Data Encryption Standard |

Table – Acronyms and Definitions