OpenSSL Cryptographic Module for Perimeta SBC

Software Module Version 1.0

FIPS 140-2 Non-Proprietary Security Policy

Document Version 1.0

Date: 2020-May-12

Prepared by:

atsec information security corporation

9130 Jollyville Road, Suite 260

Austin, TX 78759

www.atsec.com

Table of Contents

List of Tables

List of Figures

Copyright and Trademarks

Metaswitch Networks reserves the right to, without notice, modify or revise all or part of this document and/or change product features or specifications under the guidance of the Cryptography Module Validation Program (CMVP), and shall not be responsible for any loss, cost, or damage, including consequential damage, caused by reliance on these materials.

1. Introduction

1.1. Overview

This section is informative to the reader to reference to cryptographic services of OpenSSL Cryptographic Module for Perimeta SBC. Only the software listed in section 2.1 is subject to the FIPS 140-2 validation. The CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when supported if the specific operational environment is not listed on the validation certificate.

1.2. Document Overview

This Security Policy describes the features and design of the OpenSSL Cryptographic Module for Perimeta SBC using the terminology contained in the FIPS 140-2 specification. FIPS 140-2, Security Requirements for Cryptographic Module specifies the security requirements that will be satisfied by a cryptographic module utilized within a security system protecting sensitive but unclassified information. The NIST/CCCS Cryptographic Module Validation Program (CMVP) validates cryptographic module to FIPS 140-2. Validated products are accepted by the Federal agencies of both the USA and Canada for the protection of sensitive or designated information.

With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Validation Documentation is proprietary to Metaswitch Networks, Ltd. and is releasable only under appropriate non-disclosure agreements. For access to these documents, please contact the vendor.

2. Cryptographic Module Specification

This document is the non-proprietary FIPS 140-2 Security Policy for version 1.0 of the OpenSSL Cryptographic Module for Perimeta SBC (hereafter referred to as “the module”). It contains the security rules under which the module must operate and describes how this module meets the requirements as specified in FIPS PUB 140-2 (Federal Information Processing Standards Publication 140-2) for a Security Level 1 module.

The following sections describe the cryptographic module and how it conforms to the FIPS 140-2 specification in each of the required areas.

2.1. Module Overview

The OpenSSL Cryptographic Module for Perimeta SBC is a software library implementing general purpose cryptographic algorithms. The module provides cryptographic services to applications through an application program interface (API). The module also interacts with the underlying operating system via system calls.

The software block diagram in Figure 1 shows the module, its interfaces with the operational environment and the delimitation of its logical boundary.

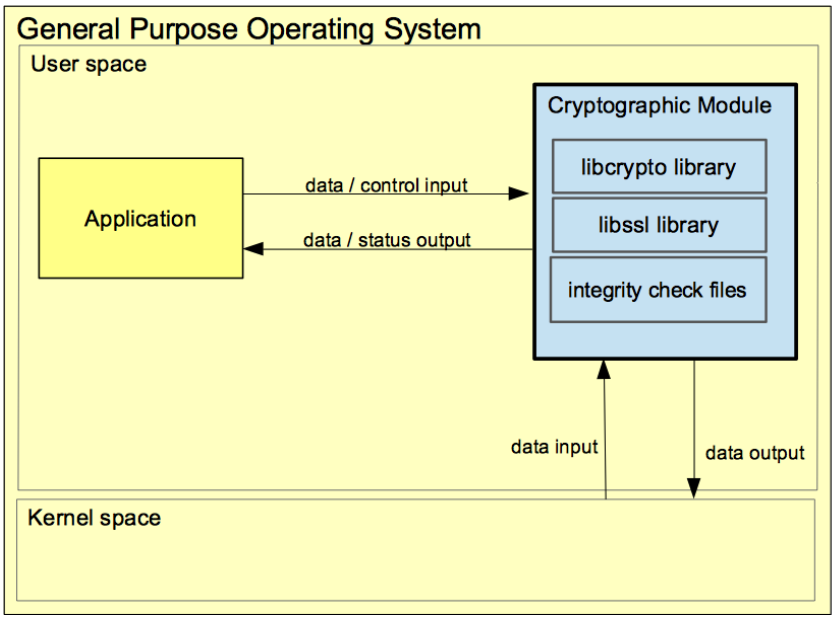

Figure 1: Software Block Diagram.

The module is implemented as a set of shared libraries. The cryptographic logical boundary consists of all shared libraries and the integrity check files used for integrity tests. Table 1 enumerates the files that comprise each module variant.

Table 1: Cryptographic Module Components.

Filename	Purpose
libssl.so.1.0.2k	Shared library for SSL protocol.
libcrypto.so.1.0.2k	Shared library for crypto primitives.

Filename	Purpose
.libssl.so.1.0.2k.hmac	Integrity check signature for SSL shared library.
.libcrypto.so.1.0.2k.hmac	Integrity check signature for crypto shared library.

The module is aimed to run in a general-purpose computer; the physical boundary is the surface of the case of the target platform. They physical boundary and its components is shown with dotted lines in Figure 2.

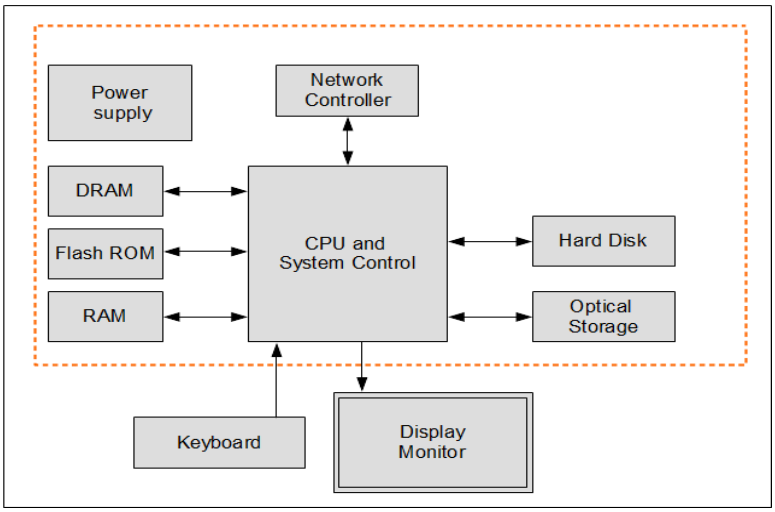

Figure 2: Cryptographic Module Physical Boundary.

2.2. FIPS 140-2 Validation

For the purpose of the FIPS 140-2 validation, the module is a software-only, multi-chip standalone cryptographic module validated at overall security level 1. Table 2 shows the security level claimed for each of the eleven sections that comprise the FIPS 140-2 standard.

Table 2: Security Levels.

FIPS 140-2 Section		Security Level
1	Cryptographic Module Specification	1
2	Cryptographic 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

FIPS 140-2 Section		Security Level
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	1
Overall Level		1

Table 3 lists the platform on which the module was tested by the laboratory, and the vendor-affirmed platforms that were tested by the vendor, indicated by the corresponding sub-titles. The table brings the corresponding module variants, configuration options and whether the platform includes Processor Algorithm Acceleration (PAA), or not (i.e., with or without PAA).

Table 3: Tested Platforms.

Hardware	Processor	Operating System	Processor Algorithm Acceleration (PAA)
Tested by the Laboratory			
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6 on VMWare Hypervisor ESXi 6.7	AES-NI
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6 on VMWare Hypervisor ESXi 6.7	None
Vendor-Affirmed Platforms			
Dell PowerEdge R620	Intel® Xeon® E5	Metaswitch Linux 6	AES-NI
Dell PowerEdge R620	Intel® Xeon® E5	Metaswitch Linux 6	None
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6	AES-NI
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6	None
Dell PowerEdge R640	Intel® Xeon® Gold	Metaswitch Linux 6	AES-NI
Dell PowerEdge R640	Intel® Xeon® Gold	Metaswitch Linux 6	None
Dell PowerEdge R720	Intel® Xeon® E5	Metaswitch Linux 6	AES-NI

Hardware	Processor	Operating System	Processor Algorithm Acceleration (PAA)
Dell PowerEdge R720	Intel® Xeon® E5	Metaswitch Linux 6	None
Dell PowerEdge R730	Intel® Xeon® E5	Metaswitch Linux 6	AES-NI
Dell PowerEdge R730	Intel® Xeon® E5	Metaswitch Linux 6	None
Dell PowerEdge R740	Intel® Xeon® Gold	Metaswitch Linux 6	AES-NI
Dell PowerEdge R740	Intel® Xeon® Gold	Metaswitch Linux 6	None
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6 on OpenStack Mitaka Hypervisor	AES-NI
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6 on OpenStack Mitaka Hypervisor	None
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6 on OpenStack Queens Hypervisor	AES-NI
Dell PowerEdge R630	Intel® Xeon® E5	Metaswitch Linux 6 on OpenStack Queens Hypervisor	None
Dell PowerEdge R720	Intel® Xeon® E5	Metaswitch Linux 6 on VMWare Hypervisor ESXi 6.7	AES-NI
Dell PowerEdge R720	Intel® Xeon® E5	Metaswitch Linux 6 on VMWare Hypervisor ESXi 6.7	None
AWS EC2 c4.2xlarge	Intel® Xeon® E5	Metaswitch Linux 6	AES-NI

Note: Per IG G.5, the CMVP makes no statement as to the correct operation of the module or the security strengths of the generated keys when the module is ported to the vendor affirmed platforms that are not listed on the validation certificate.

2.3. Modes of operation

The module supports two modes of operation:

- in "**FIPS mode**" (the FIPS Approved mode of operation), only approved or allowed security functions with sufficient security strength can be used.

- in "**non-FIPS mode**" (the non-Approved mode of operation), only non-approved security functions can be used.

The module enters FIPS mode after power-up tests succeed. Once the module is operational, the mode of operation is implicitly assumed depending on the security function invoked and the security strength of the cryptographic keys.

Critical security parameters used or stored in FIPS mode are not used in non-FIPS mode, and vice versa.

3. Cryptographic Module Ports and Interfaces

As a software-only module, the module does not have physical ports. For the purpose of the FIPS 140-2 validation, the physical ports are interpreted to be the physical ports of the hardware platform on which it runs.

The logical interfaces are the application program interface (API) through which applications request services. Table 4 summarizes the logical interfaces.

Table 4: Ports and Interfaces.

Logical Interface	Description
Data Input	API input parameters for data.
Data Output	API output parameters for data.
Control Input	API function calls.
Status Output	API return codes, error messages.
Power Input	Not applicable for the software module.

4. Roles, Services and Authentication

4.1. Roles

The module supports the following roles:

- **User role**: performs all services (in both FIPS mode and non-FIPS mode of operation), except module installation and configuration.
- **Crypto Officer role**: performs module installation and configuration.

The User and Crypto Officer roles are implicitly assumed depending on the service invoked by the entity accessing the module.

4.2. Services and Algorithms

The module provides services to calling applications that assume the User role, and human operators assuming the Crypto Officer role. Table 5 and Table 6 depict these services.

4.2.1. Services in the FIPS-Approved Mode of Operation

Table 5 lists the Approved services and the non-Approved but allowed services in FIPS mode of operation, the roles that can request the service, the algorithms involved with their corresponding CAVP certificate numbers (if applicable), the Critical Security Parameters involved and how they are accessed.

The following convention is adopted when specifying the access permissions that the service has for each CSP or key.

- **Create**: the service can create a new CSP.
- **Read**: the service can read an existing CSP.
- **Update**: the service can write a new value to an existing CSP.
- **Zeroize**: the service can zeroize the existing CSP.
- **N/A (not applicable)**: the service does not access any CSP during its operation.

Table 5: Cryptographic Services in FIPS mode of operation.

Service	Description and Algorithms	Role	Access	CSP
Symmetric Encryption and Decryption	AES	User	Read	AES key
	Triple-DES	User	Read	Triple-DES key
Asymmetric Key Generation	RSA, DSA/Diffie-Hellman, ECDSA/EC Diffie-Hellman	User	Create	RSA private key
Message digest	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	User	n/a	none
Message Authentication Code (MAC)	HMAC, CMAC with AES	User	Read	HMAC key AES Key

Service	Description and Algorithms	Role	Access	CSP
Random Number Generation	Hash_DRBG, HMAC_DRBG, CTR_DRBG	User	Read, Update	DRBG Entropy Input and Internal state
Digital signature generation and verification	RSA, DSA, ECDSA	User	Read	Asymmetric Private key
DSA domain parameter generation and verification	DSA	User	Read	DSA domain parameters
ECDSA public key verification	ECDSA	User	Read	ECDSA private key
Key Wrapping	AES KW, AES KWP, RSA	User	Read	AES Key, RSA private Key
Key Derivation Function	SP 800-135 TLS 1.0/1.1, TLS 1.2 KDF	User	Read	Shared secret and Derived Key
TLS network protocol	Provide data encryption and authentication over TLS network protocol with AES, Triple-DES, HMAC.	User	Create, Read, Zeroize	AES, Triple-DES, HMAC key
TLS Key agreement	AES, Triple-DES, HMAC, RSA/DSA/ECDSA, Diffie-Hellman, EC Diffie-Hellman	User	Create, Read	AES, Triple-DES keys, RSA, DSA, ECDSA private keys, HMAC key, Shared Secret, Diffie-Hellman or EC Diffie-Hellman private Keys
Show status	Show status of the module state	User	N/A	none
Self-Tests	Initiate power-on self-tests	User	N/A	none[1]
Zeroization	Zeroize all critical security parameters	User	Zeroize	All CSPs
Module Installation	Installation of the module	Crypto Officer	N/A	none
Module Configuration	Configuration of the module	Crypto Officer	N/A	none

[1] The HMAC key used for integrity test is not considered a critical security parameter.

4.2.2. Services in the non-FIPS-Approved Mode of Operation

Table 6 lists the services only available in non-FIPS mode of operation.

Table 6: Services in non-FIPS mode of operation.

Service	Description and Algorithms	Role
Symmetric encryption and decryption	using non-approved algorithms, such as Blowfish, Camellia, CAST, DES, IDEA, RC2, RC4, RC5, SEED.	User
Asymmetric key generation	using RSA, DSA, Diffie-Hellman, ECDSA, EC Diffie-Hellman keys/curves not listed in Table 7 or Table 8.	User
Digital signature generation	using RSA, DSA, ECDSA keys not listed in Table 7 or generation using SHA-1.	User
Message digest generation	using non-approved algorithms, such as MD2, MD4, MD5, MDC-2, RIPEMD160, Whirlpool.	User
MAC generation and verification	using Triple-DES.	User
Random Number Generation	using ANSI X9.31 RNG.	User
Key Wrapping	using RSA keys not listed in Table 7 or Table 8.	User
TLS key agreement	using Diffie-Hellman, EC Diffie-Hellman and RSA keys/curves not listed in Table 7 or Table 8.	User
J-PAKE Key Agreement	password authenticated key agreement using J-PAKE.	User

4.2.3. FIPS-Approved Algorithms

Table 7 shows the FIPS-approved cryptographic algorithms, available in the FIPS mode of operation. Separate entries distinguish the cryptographic algorithms implemented using Processor Algorithm Acceleration (PAA).

Table 7: FIPS-Approved Cryptographic Algorithms.

CAVP Certificate	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli (in bits)	Use
SSSE3 #C1007	DRBG	[SP800-90A]	Hash_DRBG SHA-1, SHA-224, SHA-256 with/without PR	n/a	Random Number Generation
			HMAC_DRBG HMAC with SHA-1, SHA-224, SHA-256 with/without PR		

CAVP Certificate	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli (in bits)	Use
	HMAC	[FIPS198-1]	SHA-1, SHA-224, SHA-256	112 bits or greater	Message Authentication Code
	SHS	[FIPS180-4]	SHA-1, SHA-224, SHA-256	n/a	Message Digest
VPAES #C1008	AES	[FIPS197], [SP800-38A]	ECB, CBC, OFB, CFB1, CFB8, CFB128, CTR	128, 192, 256	Data Encryption and Decryption
		[FIPS197], [SP800-38B]	CMAC	128, 192, 256	Message Authentication Code
		[FIPS197], [SP800-38C]	CCM	128, 192, 256	Authenticated Encryption and Decryption
		[FIPS197], [SP800-38D]	GCM	128, 192, 256	Authenticated Encryption and Decryption
		[FIPS197], [SP800-38E]	XTS	128, 256	Data Encryption and Decryption
		[FIPS197], [SP800-38F]	KW, KWP	128, 192, 256	Key Wrapping and Unwrapping
	DRBG	[SP800-90A]	CTR_DRBG AES-128, AES-192, AES-256 with/without PR, with/without DF	n/a	Random Number Generation
AES-NI/ SHA AVX #C1009	AES	[FIPS197], [SP800-38A]	ECB, CBC, OFB, CFB1, CFB8, CFB128, CTR	128, 192, 256	Data Encryption and Decryption
		[FIPS197], [SP800-38B]	CMAC	128, 192, 256	Message Authentication Code
		[FIPS197], [SP800-38C]	CCM	128, 192, 256	Authenticated Encryption and Decryption

CAVP Certificate	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli (in bits)	Use
		[FIPS197], [SP800-38D]	GCM	128, 192, 256	Authenticated Encryption and Decryption
		[FIPS197], [SP800-38E]	XTS	128, 256	Data Encryption and Decryption
		[FIPS197], [SP800-38F]	KW, KWP	128, 192, 256	Key Wrapping and Unwrapping
	DRBG	[SP800-90A]	CTR_DRBG AES-128, AES-192, AES-256 with/without PR, with/without DF	n/a	Random Number Generation
			Hash_DRBG SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) with/without PR		
			HMAC_DRBG HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 with/without PR		
	DSA	[FIPS186-4]	n/a	L=2048, N=224; L=2048, N=256; L=3072, N=256	Key Generation
			SHA-224, SHA-256, SHA-384, SHA-512	L=2048, N=224;	Domain Parameter Generation Domain Parameter Verification
			SHA-256, SHA-384, SHA-512	L=2048, N=256; L=3072, N=256	
			SHA-224, SHA-256, SHA-384, SHA-512	L=2048, N=224; L=2048, N=256; L=3072, N=256	Signature Generation Signature Verification

CAVP Certificate	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli (in bits)	Use
	ECDSA	[FIPS186-4]	Testing Candidates	P-256, P-384, P-521	Key Generation
			n/a	P-256, P-384, P-521	Public Key Verification
			SHA-224, SHA-256, SHA-384, SHA-512	P-256, P-384, P-521	Signature Generation
			SHA1, SHA-224, SHA-256, SHA-384, SHA-512	P-256, P-384, P-521	Signature Verification
	HMAC	[FIPS198-1]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 bits or greater	Message Authentication Code
	KAS ECC	[SP800-56A]	ECC Ephemeral Unified Scheme	P-256 (EC), P-384 (ED), P-521 (EE)	Elliptic Curve Diffie-Hellman Shared Secret Computation
	KAS FFC	[SP800-56A]	FFC dhEphem Scheme	p=2048, q=224 (FB); p=2048, q=256 (FC)	Diffie-Hellman Shared Secret Computation
	TLS 1.0/1.1/1.2	[SP800-135]	SHA-256, SHA-384	n/a	Key Derivation
	RSA	[FIPS186-4]	B.3.3 Random Probable Prime	2048, 3072	Key Generation
			X9.31 with SHA-256, SHA-384, SHA-512	2048, 3072	Signature Generation
			PKCS#1 v1.5 and PSS with SHA-224, SHA-256, SHA-384, SHA-512		

CAVP Certificate	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli (in bits)	Use
			X9.31 with SHA-1, SHA-256, SHA-384, SHA-512	1024, 2048, and 3072	Signature Verification
			PKCS#1 v1.5 and RSA PSS with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		
		[FIPS186-2]	X9.31 with SHA-256, SHA-384, SHA-512	4096	Signature Generation
			PKCS#1 v1.5 and PSS with SHA-224, SHA-256, SHA-384, SHA-512	4096	
	SHS	[FIPS180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512,	n/a	Message Digest
	Triple-DES	[SP800 67], [SP800-38A]	ECB, CBC, CFB-1, CFB8, CFB64, OFB, CTR	192	Data Encryption and Decryption
Assembler #C1010	AES	[FIPS197], [SP800-38A]	ECB, CBC, OFB, CFB1, CFB8, CFB128, CTR	128, 192, 256	Data Encryption and Decryption
		[FIPS197], [SP800-38B]	CMAC	128, 192, 256	Message Authentication Code
		[FIPS197], [SP800-38C]	CCM	128, 192, 256	Authenticated Encryption and Decryption
		[FIPS197], [SP800-38D]	GCM	128, 192, 256	Authenticated Encryption and Decryption

CAVP Certificate	Algorithm	Standard	Mode/Method	Key Lengths, Curves or Moduli (in bits)	Use
		[FIPS197], [SP800-38E]	XTS	128, 256	Data Encryption and Decryption
		[FIPS197], [SP800-38F]	KW, KWP	128, 192, 256	Key Wrapping and Unwrapping
	DRBG	[SP800-90A]	CTR_DRBG AES-128, AES-192, AES-256 with/without PR, with/without DF	n/a	Random Number Generation
			Hash_DRBG SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) with/without PR		
			HMAC_DRBG HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 with/without PR		
	HMAC	[FIPS198-1]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	112 bits or greater	Message Authentication Code
	SHS	[FIPS180-4]	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512,	n/a	Message Digest

4.2.4. Non-FIPS-Approved Algorithms

Table 8 lists the cryptographic algorithms that are non-approved but allowed in the FIPS mode of operation.

Table 8: FIPS-Allowed Cryptographic Algorithm.

Algorithm	Caveat	Use
Diffie-Hellman with key size between 2048 bits and 10000 bits	Provides between 112 and 220 bits of encryption strength.	Key Establishment

Algorithm	Caveat	Use
EC Diffie-Hellman with P-256, P-384, P-521 curves	Provides between 128 and 256 bits of encryption strength.	Key Establishment
RSA Key Wrapping with key size between 2048 bits and 15360 bits (or more)	Provides between 112 and 256 bits of encryption strength.	Key Establishment
MD5	N/A	Message digest used in TLS 1.0/1.1 only.
NDRNG	N/A	Seeding for the DRBG.

Table 9 shows the cryptographic algorithms implemented in the module that are not allowed in FIPS mode of operation.

Table 9: Non-Approved Cryptographic Algorithms.

Algorithm	Use
Blowfish, Camellia, CAST, DES, IDEA, RC2, RC4, RC5, SEED	Data Encryption/Decryption
MD2, MD4, MD5, MDC-2, RIPEMD160, Whirlpool	Message Digest
ANSI X9.31 RNG	Random number generation
RSA	Key generation/Signature generation with keys of length not listed in Table 7 or Table 8.
Diffie-Hellman	Key agreement using keys of length not listed in Table 7 or Table 8.
DSA	Parameter/Key generation/Signature generation with keys of length not listed in Table 7 or Table 8.
EC Diffie-Hellman	Key agreement using curves not listed in Table 7 or Table 8.
ECDSA	Key generation/Signature generation with curves not listed in Table 7 or Table 8.
SHA-1	Signature generation
J-PAKE	Password Authenticated Key Exchange

4.3. Operator Authentication

The module does not implement user authentication. The role of the user is implicitly assumed based on the service requested.

5. Physical Security

The module is comprised of software only and therefore this security policy does not make any claims on physical security.

6. Operational Environment

6.1. Applicability

The module operates in a modifiable operational environment per FIPS 140-2 level 1 specifications. The module runs on a commercially available general-purpose operating system executing on the hardware specified in section 2.2.

6.2. Policy

The operating system is restricted to a single operator; concurrent operators are explicitly excluded. The application that requests cryptographic services is the single user of the module.

7. Cryptographic Key Management

Table 10 summarizes the Critical Security Parameters (CSPs) that are used by the cryptographic services implemented in the module. All CSPs are stored in RAM.

Table 10: Life cycle of Critical Security Parameters (CSP).

Name	Generation	Entry/Output	Zeroization
AES keys	Provided by the calling application, or derived during TLS handshake using SP800-135 KDF.	Entry: via API input parameter. Output: No output.	EVP_CIPHER_CTX_cleanup()
Triple-DES keys	Provided by the calling application, or derived during TLS handshake using SP800-135 KDF.	Entry: via API input parameter. Output: No output.	EVP_CIPHER_CTX_cleanup()
HMAC key	Provided by the calling application, or derived during TLS handshake using SP800-135 KDF.	Entry: via API input parameter. Output: No output.	HMAC_CTX_cleanup()
RSA key pair	Keys are generated using FIPS 186-4 and the random value used in the key generation is obtained from SP800- 90A DRBG.	Entry: via API input parameter in plaintext. Output: via API output parameters in plaintext.	RSA_free()
DSA key pair			DSA_free()
ECDSA key pair			EC_KEY_free()
Diffie-Hellman key pair	Keys are generated using FIPS 186-4 and the random value used in the key generation is obtained from SP800- 90A DRBG.	Entry: via API input parameter in plaintext. Output: via API output parameters in plaintext.	DH_free()
EC Diffie-Hellman key pair			EC_KEY_free()
Shared secret (pre-master secret)	Generated during the key agreement when using Diffie-Hellman or EC Diffie-Hellman key exchange. Generated by TLS client as output from DRBG when using RSA key exchange.	Entry: if received by module as TLS server, wrapped with server's public RSA key; otherwise no entry. Output: if generated by module as TLS client, wrapped with server's public RSA key; otherwise, no output.	SSL_free() and SSL_clear()
Entropy input string (seed)	Obtained from NDRNG	N/A	FIPS_drbg_free()
DRBG internal state (V, C, Key)	During DRBG initialization	N/A	FIPS_drbg_free()

The following sections describe how CSPs, in particular cryptographic keys, are managed during their life cycle.

7.1. Random Number Generation

The module employs a DRBG based on [SP800-90A] for the creation of asymmetric keys, and for providing an RNG service to calling applications.

The DRBG supports the Hash_DRBG, HMAC_DRBG and CTR_DRBG mechanisms. The DRBG is initialized during module initialization; the module loads by default the AES-256 CTR_DRBG.

The DRBG is initialized during module initialization and seeded with the NDRNG from /dev/urandom. The NDRNG is provided by the operational environment (i.e., Linux RNG), which is within the module's physical boundary but outside of the module's logical boundary. The NDRNG from the Linux RNG provides, at its output, the seed for the DRBG in the form of a 384-bit entropy_input string. The output of the NDRNG provides at least 256 bits of entropy in its 384 bits of length that is then used for the DRBG initialization (seeding) and reseeding. The entropy is sufficient for the security strength provided by the DRBG algorithm.

The module performs DRBG health tests as defined in Section 11.3 of [SP800-90A], and continuous random number generator tests (CRNGT) on the output of the SP800-90A DRBG to ensure that consecutive random numbers do not repeat. The operational environment (the Linux RNG) performs the continuous test on the NDRNG.

7.2. Key Generation

For generating RSA, DSA and ECDSA keys the module implements asymmetric key generation services compliant with [FIPS186-4] and using a DRBG compliant with [SP800-90A]. The random value used in asymmetric key generation is obtained from the DRBG. The public and private key pairs used in the Diffie-Hellman and EC Diffie-Hellman key agreement schemes are generated internally by the module using the same DSA and ECDSA key generation mechanisms compliant with [FIPS186-4] and [SP800-56A].

The module does not provide a dedicated service for symmetric key generation. Symmetric keys are derived from the shared secret established by Diffie-Hellman and EC Diffie-Hellman in a manner that is compliant to NIST SP 800-135 for TLS KDF.

In accordance with FIPS 140-2 IG D.12, the cryptographic module performs Cryptographic Key Generation (CKG) for asymmetric keys as per SP800-133 (vendor affirmed).

7.3. Key Establishment

The module provides Diffie-Hellman and EC Diffie-Hellman key agreement schemes used as part of the TLS protocol key exchange. The module also provides AES key wrapping per [SP800-38F] and RSA key wrapping using public key encryption and private key decryption primitives as allowed by [FIPS140-2_IG] D.9. RSA key wrapping may be used as part of the TLS protocol key exchange.

- AES key wrapping provides between 128 and 256 bits of encryption strength.
- RSA key wrapping provides between 112 and 256 bits of encryption strength.
- Diffie-Hellman key establishment methodology provides between 112 and 220 bits of encryption strength.
- EC Diffie-Hellman key establishment methodology provides between 128 and 256 bits of encryption strength.

7.4. Key Entry/Output

The module does not support manual key entry or intermediate key generation key output. In addition, the module does not produce key output in plaintext format outside its physical boundary. The keys can be entered or output from the module in plaintext form via API parameters, to and from the calling application only.

7.5. Key/CSP Storage

Public and private keys are provided to the module by the calling process, and are destroyed when released by the appropriate API function calls. The module does not perform persistent storage of keys. The only exception is the HMAC key used for integrity test, which is stored in the module and relies on the operating system for protection.

7.6. Key/CSP Zeroization

The memory occupied by keys is allocated by regular memory allocation operating system calls. The application is responsible for calling the appropriate destruction functions provided in the module's API. The destruction functions overwrite the memory occupied by keys with “zeros” and deallocates the memory with the regular memory deallocation operating system call.

8. Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC)

The test platforms listed in Table 3 have been tested and found to conform to the EMI/EMC requirements specified by 47 Code of Federal Regulations, FCC PART 15, Subpart B, Unintentional Radiators, Digital Devices, Class A (i.e., Business use). These devices are designed to provide reasonable protection against harmful interference when the devices are operated in a commercial environment.

9. Self Tests

9.1. Power-Up Tests

The module performs power-up tests automatically making use of default entry point (DEP) without any operator intervention when the module is loaded into memory; power-up tests ensure that the module is not corrupted and that the cryptographic algorithms work as expected.

While the module is executing the power-up tests, services are not available, and input and output are inhibited. The module does not return control to the calling application until the power-up tests are completed. On successful completion of the all the power-up tests, the module becomes operational and crypto services are then available. If any of the tests fails, the module transitions to the error state and subsequent calls to the module will fail. Thus, in the error state, no further cryptographic operations will be possible.

9.1.1. Integrity Tests

The integrity of the module is verified by comparing an HMAC-SHA-256 value calculated at run time with the HMAC value stored in the module that was computed at build time.

9.1.2. Cryptographic algorithm tests

The module performs self-tests on all FIPS-Approved cryptographic algorithms supported in the approved mode of operation, using the known answer tests (KAT) or a pairwise consistency test (PCT) (Table 11).

Table 11: Self-Tests.

Algorithm	Test
AES	• KAT AES-GCM with 256-bit key, encryption • KAT AES-GCM with 256-bit key, decryption • KAT AES-ECB with 128-bit key, encryption • KAT AES-ECB with 128-bit key, decryption • KAT AES-CCM with 192-bit key, encryption • KAT AES-CCM with 192-bit key, decryption • KAT AES-XTS with 128-bit and 256-bit keys, encryption • KAT AES-XTS with 128-bit and 256-bit keys, decryption • KAT AES-CMAC with 128-bit, 192-bit, and 256-bit key
DRBG	• KAT CTR_DRBG with AES-256, with and without DF, with and without PR • KAT Hash_DRBG with SHA-256 with and without PR • KAT HMAC_DRBG with HMAC-SHA-256 with and without PR
DSA	• PCT DSA with L=2048, N=224 and SHA-256
ECDSA	• PCT ECDSA with P-256 and SHA-256
HMAC	• KAT HMAC-SHA-1 • KAT HMAC-SHA-224 • KAT HMAC-SHA-256

Algorithm	Test
	• KAT HMAC-SHA-384 • KAT HMAC-SHA-512
KAS ECC (EC Diffie-Hellman)	• Primitive “Z” Computation KAT with P-256 curve
KAS FFC (Diffie-Hellman)	• Primitive “Z” Computation KAT with 2048-bit key
RSA	• KAT RSA 2048-bit key (PKCS#1 v1.5, PSS) with SHA-224, SHA-256, SHA-384 and SHA-512, signature generation • KAT RSA 2048-bit key (PKCS#1 v1.5, PSS) with SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512, signature verification
SHS	• KAT SHA-1 • KAT SHA-256 • KAT SHA-512
Triple DES	• KAT Triple-DES-ECB with 192-bit key, encryption • KAT Triple-DES-ECB with 192-bit key, decryption

9.2. On-Demand self-tests

The module provides the Self-Test service to perform self-tests on demand. The on demand Self-Tests can also be invoked by powering-off and reloading the module. This service performs the same cryptographic algorithm tests executed during power-up. During the execution of the on-demand self-tests, services are not available and no data output or input is possible.

9.3. Conditional Tests

The module performs conditional tests on the cryptographic algorithms shown in Table 12.

Table 12: Conditional Tests.

Algorithm	Test
DSA key generation	• Pair-wise consistency test
ECDSA key generation	• Pair-wise consistency test
RSA key generation	• Pair-wise consistency test
DRBG	• Continuous Random Number Generator Test (CRNGT)

10. Guidance

10.1. Crypto Officer Guidance

The module is distributed alongside the remainder of the Perimeta software and operating system that are installed onto Metaswitch's Perimeta SBC. The software is distributed in different ways depending on the customer requirements. Customers will receive detailed instructions for acquiring and installing the secure software.

For the proper installation and configuration of the module, this guidance details two cases: (1) the full installation of a Perimeta SBC image that includes the secure module; and (2) upgrading a previous version of the Perimeta SBC to the version that includes the secure module. These cases are described next.

10.1.1. Full Installation of Perimeta SBC Image

This case comprises the installation of a version of Perimeta SBC that includes the secure module following the normal processes. Here, the module is already properly configured from the vendor, and there is no additional configuration required from the Crypto Officer for the install process. The Crypto Officer simply installs the appropriate image and then the installation and configuration are considered complete.

10.1.2. Upgrade Installation from another Version of Perimeta SBC

A previous version of the Perimeta SBC can be upgraded to the version that includes the secure module. The specific upgrade instructions are provided with the Perimeta SBC package, and involves first terminating the Perimeta software and decommissioning the system, and then proceeding with the upgrade.

After the upgrade is performed, the secure module is present in a packaged form within the Perimeta SBC. To use the module as the validated module with its modes of operation per the rules in Section 2.3, the FIPS capabilities of the module must be enabled as such:

1. The Crypto Officer is required to log in to the Perimeta SBC environment (using a physical console, an emulated physical console, or through SSH).
2. In the menu that is presented, the Crypto Officer selects the option to enable FIPS capabilities. Once the FIPS capabilities are enabled, the system automatically performs the following actions:
    a. Records that the FIPS capabilities are enabled and ready to use.
    b. Removes the unpacked versions of OpenSSL and OpenSSH from disk.
    c. Adds the appropriate kernel boot parameters to ensure that the FIPS capabilities are enabled upon boot.
    d. Reboots the operational environment to have the changes take effect.

After the reboot, the installation and configuration process of the module is complete.

10.2. User Guidance

To run the module in FIPS mode, the user shall only use the FIPS approved or allowed services listed in Table 5. The function calls FIPS_mode_set(0), ENGINE_register_* and ENGINE_set_default_* are prohibited while running the module.

10.2.1. Random Number Generator

The OpenSSL API call of RAND_cleanup must not be used. This call will clean up the internal DRBG state. This call also replaces the DRBG instance with the non-FIPS Approved SSLeay Deterministic Random Number Generator when using the RAND_* API calls.

10.2.2. AES GCM IV

AES GCM encryption and decryption shall only be used in the context of the TLS protocol version 1.2 for compliance with IG A.5, Scenario 1 [FIPS140-2_IG]. The module is compliant with [SP 800-52r2] and the mechanism for IV generation is compliant with [RFC5288]. The operations of one of the two parties involved in the TLS key establishment scheme are performed entirely within the cryptographic boundary of the module, including the setting of the counter portion of the IV.

The nonce_explicit part of the IV does not exhaust the maximum number of possible values for a given session key. The design of the TLS protocol in this module implicitly ensures that the nonce_explicit, or counter portion, of the IV will not exhaust all of its possible values, per the mechanisms in Section 7.4.1.1 and Section 7.4.1.2 in [RFC5246] and compliant to IG A.5.

In case the module's power is lost and then restored, the key used for AES GCM encryption or decryption shall be re-distributed

10.2.3. Triple-DES Data Encryption

Data encryption using the same three-key Triple-DES key shall not exceed 2^{16} Triple-DES (64-bit) blocks, in accordance to [SP800-67] and IG A.13 in [FIPS140-2-IG].

10.2.4. AES-XTS

The AES algorithm in XTS mode can be only used for the cryptographic protection of data on storage devices, as specified in [SP800-38E]. The length of a single data unit encrypted with the XTS-AES shall not exceed 2^{20} AES blocks. In addition, to meet the requirement in [FIPS140-2_IG] A.9, the module implements a check to ensure that the two AES keys used in XTS-AES algorithm are not identical.

10.3. Handling Self-test Errors

The module transition to the error state when any of self-tests or conditional tests fails. The application must be restarted to recover from these errors. Following are the error messages specific to self-test failure:

FIPS_R_FINGERPRINT_DOES_NOT_MATCH – The integrity verification check failed

FIPS_R_SELFTEST_FAILED – a known answer test failed

FIPS_R_TEST_FAILURE – a known answer test failed (RSA); pairwise consistency test failed (DSA)

FIPS_R_PAIRWISE_TEST_FAILED – a pairwise consistency test failed during EC/DSA or RSA key generation

FIPS_R_DRBG_STUCK – the DRBG generated two same consecutive values

These errors are reported through the regular ERR interface of the module and can be queried by functions such as ERR_get_error(). See the OpenSSL manual page for the function description.

When the module is in error state, output is inhibited and no crypto operations are available. Any calls to the crypto functions in error state will return error message: 'FATAL FIPS SELFTEST FAILURE' on stderr and the application is terminated with the abort() call.

The only way to recover from the error state is to reload the module and restart the application. If failures persist, the module must be reinstalled.

10.4. Enabling Mitigation of Other Attacks

To enable the mechanisms implemented in the module to mitigate other attacks, please refer to Section 11.

11. Mitigation of Other Attacks

RSA is vulnerable to timing attacks. In a setup where attackers can measure the time of RSA decryption or signature operations, blinding must be used to protect the RSA operation from that attack. The API function of RSA_blinding_on turns blinding on for the RSA key and generates a random blinding factor. The random number generator must be seeded prior to calling RSA_blinding_on.

Weak Triple-DES keys are detected per the code excerpt below.

```
/* Weak and semi week keys as taken from
* %A D.W. Davies
* %A W.L. Price
* %T Security for Computer Networks
* %I John Wiley & Sons
* %D 1984
* Many thanks to smb@ulysses.att.com (Steven Bellovin) for the reference
* (and actual cblock values).
*/
#define NUM_WEAK_KEY 16
static const DES_cblock weak_keys[NUM_WEAK_KEY]={
/* weak keys */
{0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},
{0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE},
{0x1F,0x1F,0x1F,0x1F,0x0E,0x0E,0x0E,0x0E},
{0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},
/* semi-weak keys */
{0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},
{0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},
{0x1F,0xE0,0x1F,0xE0,0x0E,0xF1,0x0E,0xF1},
{0xE0,0x1F,0xE0,0x1F,0xF1,0x0E,0xF1,0x0E},
{0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},
{0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01},
{0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E,0xFE},
{0xFE,0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E},
{0x01,0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E},
{0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E,0x01},
{0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1,0xFE},
{0xFE,0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1}};
```

Please note that there is no weak key detection by default. The caller can explicitly set the DES_check_key to 1 or call DES_check_key_parity() and/or DES_is_weak_key() functions on its own.

Appendix A. Glossary and Abbreviations

AES	Advanced Encryption Standard
AES-NI	Advanced Encryption Standard New Instructions
CAVP	Cryptographic Algorithm Validation Program
CAVS	Cryptographic Algorithm Validation Scheme
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining-Message Authentication Code
CFB	Cipher Feedback
CMAC	Cipher-based Message Authentication Code
CMVP	Cryptographic Module Validation Program
CSP	Critical Security Parameter
CTR	Counter Mode
DF	Derivation Function
DSA	Digital Signature Algorithm
DRBG	Deterministic Random Bit Generator
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standards Publication
GCM	Galois Counter Mode
HMAC	Hash Message Authentication Code
KAS	Key Agreement Schema
KAT	Known Answer Test
KW	AES Key Wrap
KWP	AES Key Wrap with Padding
MAC	Message Authentication Code
NIST	National Institute of Science and Technology
NDRNG	Non-Deterministic Random Number Generator
OFB	Output Feedback
PAA	Processor Algorithm Acceleration
PR	Prediction Resistance
PSS	Probabilistic Signature Scheme
RNG	Random Number Generator
RSA	Rivest, Shamir, Addleman
SHA	Secure Hash Algorithm
TDES	Triple-DES

XTS	XEX-based Tweaked-codebook mode with cipher text Stealing

Appendix B. References

FIPS140-2 **FIPS PUB 140-2 - Security Requirements For Cryptographic Modules**
May 2001
http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

FIPS140-2_IG **Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program**
August 16, 2019
http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf

FIPS180-4 **Secure Hash Standard (SHS)**
March 2012
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

FIPS186-4 **Digital Signature Standard (DSS)**
July 2013
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

FIPS197 **Advanced Encryption Standard**
November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

FIPS198-1 **The Keyed Hash Message Authentication Code (HMAC)**
July 2008
http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf

PKCS#1 **Public Key Cryptography Standards (PKCS) #1: RSA Cryptography**
Specifications Version 2.1
February 2003
http://www.ietf.org/rfc/rfc3447.txt

RFC5246 **The Transport Layer Security (TLS) Protocol Version 1.2**
August 2008
https://tools.ietf.org/html/rfc5246

RFC5288 **AES Galois Counter Mode (GCM) Cipher Suites for TLS**
August 2008
https://tools.ietf.org/html/rfc5288

SP800-38A **NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation Methods and Techniques**
December 2001
http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

SP800-38B **NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication**
May 2005
http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf

SP800-38C	**NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality** May 2004 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf
SP800-38D	**NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC** November 2007 http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
SP800-38E	**NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS AES Mode for Confidentiality on Storage Devices** January 2010 http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
SP800-38F	**NIST Special Publication 800-38F - Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping** December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38F.pdf
SP800-52r2	**NIST Special Publication 800-52 Revision 2 – Guidelines for the selection, configuration, and use of Transport Layer Security (TLS) implementations** August, 2019 https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r2.pdf
SP800-56A	**NIST Special Publication 800-56A - Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised)** March, 2007 http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf
SP800-56Ar2	**NIST Special Publication 800-56A Revision 2 - Recommendation for Pair Wise Key Establishment Schemes Using Discrete Logarithm Cryptography** May 2013 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf
SP800-56B	**Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography** August 2009 http://csrc.nist.gov/publications/nistpubs/800-56B/sp800-56B.pdf
SP800-57	**NIST Special Publication 800-57 Part 1 Revision 4 - Recommendation for Key Management Part 1: General** January 2016 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

SP800-67	**NIST Special Publication 800-67 Revision 1 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher** January 2012 http://csrc.nist.gov/publications/nistpubs/800-67-Rev1/SP-800-67-Rev1.pdf
SP800-90A	**NIST Special Publication 800-90A - Revision 1 - Recommendation for Random Number Generation Using Deterministic Random Bit Generators** June 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
SP800-90B	**(Second DRAFT) NIST Special Publication 800-90B - Recommendation for the Entropy Sources Used for Random Bit Generation** January 2016 http://csrc.nist.gov/publications/drafts/800-90/sp800-90b_second_draft.pdf
SP800-108	**NIST Special Publication 800-108 - Recommendation for Key Derivation Using Pseudorandom Functions (Revised)** October 2009 http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
SP800-131A	**NIST Special Publication 800-131A Revision 1- Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths** November 2015 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf
SP800-132	**NIST Special Publication 800-132 - Recommendation for Password-Based Key Derivation - Part 1: Storage Applications** December 2010 http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf
SP800-133	**NIST Special Publication 800-133 - Recommendation for Cryptographic Key Generation** December 2012 http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-133.pdf
SP800-135	**NIST Special Publication 800-135 Revision 1 - Recommendation for Existing Application-Specific Key Derivation Functions** December 2011 http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf