Software Diversified Services

Cryptographic Module

Version 1.0.0

FIPS 140-2 Non-Proprietary Security Policy

Level 1 Validation

Document revision 016, May 2018

Software Diversified Services
1322 81st Ave NE,
Minneapolis, MN 55432

t: 763 571 9000
e: info@sdsusa.com
w: www.sdsusa.com

Prepared for SDS by

Rycombe Consulting Limited
http://www.rycombe.com
+44 1273 476366

Contents

Figures

1 Introduction

This section identifies the cryptographic module; describes the purpose of this document; provides external references for more information; and explains how the document is organized.

1.1 Identification

Module Name	SDS Cryptographic Module
Module Version	1.0.0

1.2 Purpose

This is the non-proprietary FIPS 140-2 Security Policy for the SDS Cryptographic Module, also referred to as “the module” within this document. This Security Policy details the secure operation of SDS Cryptographic Module as required in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

1.3 References

For more information on SDS products please visit: www.sdsusa.com. For more information on NIST and the Cryptographic Module Validation Program (CMVP), please visit http://csrc.nist.gov/groups/STM/cmvp/index.html.

1.4 Document Organization

This Security Policy document is one part of the FIPS 140-2 Submission Package. This document outlines the functionality provided by the module and gives high-level details on the means by which the module satisfies FIPS 140-2 requirements. With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Submission documentation may be SDS proprietary or otherwise controlled and releasable only under appropriate non-disclosure agreements. For access to these documents, please contact SDS.

The various sections of this document map directly onto the sections of the FIPS 140-2 standard and describe how the module satisfies the requirements of that standard.

1.5 Document Terminology

Term	Description
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
API	Application Programming Interface
BIOS	Basic Input Output Services
CAVP	Cryptographic Algorithm Validation Program
CMSP	Cryptographic Module Security Policy
CMVP	Cryptographic Module Validation Program
CPU	Central Processing Unit (Microprocessor)
CSP	Critical Security Parameters
DES	Data Encryption Standard
DRBG	Deterministic Random-bit Generator
DVD	Digital Video Disc
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
HDD	Hard Disk Drive
HMAC	Keyed-Hash Message Authentication Code
KDF	Key Derivation Function
LCD	Liquid Crystal Display
LED	Light Emitting Diode
N/A	Not Applicable
NDRNG	Non-deterministic Random Number Generator
NIST	National Institute of Standards and Technology
OS	Operating System
RAM	Rndom-access Memory
RBG	Random Bit Generator
RFC	Request for Comments
RNG	Random Number Generator
RSA	An algorithm for public-key cryptography. Named after Rivest, Shamir and Adleman who first publicly described it.
SCSI	Small Computer System Interface
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
SP	NIST Special Publication document
TLS	Transport Layer Security
Triple-DES	Triple-DES
USB	Universal Serial Bus

Figure 1 Document terminology

2 SDS Cryptographic Module

This section provides the details of how the module meets the FIPS 140-2 requirements.

2.1 Overview

The module provides cryptographic services to SDS products.

2.2 Module Specification

The SDS Cryptographic Module is a software module that provides cryptographic services to SDS products.

The module is classified as a multi-chip standalone module.

The module provides a number of NIST validated cryptographic algorithms. The module provides applications with a library interface that enables them to access the various cryptographic algorithm functions supplied by the module.

2.2.1 Hardware, Software and Firmware components

The module is a software module that resides on the hardware of a general-purpose computer (see Figure 4). For the purposes of FIPS 140-2 testing, the module is evaluated running on the operational environments defined in section 2.6.

The module is packaged as a number of distinct binary images:

OPERATIONAL ENVIRONMENT	FILE NAME(S)
Windows	SDSEngine.dll
Linux	libenc_eng_fips.so.1.0.0
AIX	libenc_eng_fips.so.1.0.0

Figure 2 Module binary images

2.2.2 Cryptographic Boundary

The cryptographic boundary of the module is the case of the general-purpose computer (GPC) on which it is installed. See Figure 3. The module is a software module running in a well-defined operational environment on a general-purpose computer. The processor of this platform executes all software. All software components of the module are persistently stored within the device and, while executing, are stored in the device local RAM.

The cryptographic boundary of the module is shown in Figure 4. The only software within the logical boundary of the cryptographic boundary is listed in Figure 2.

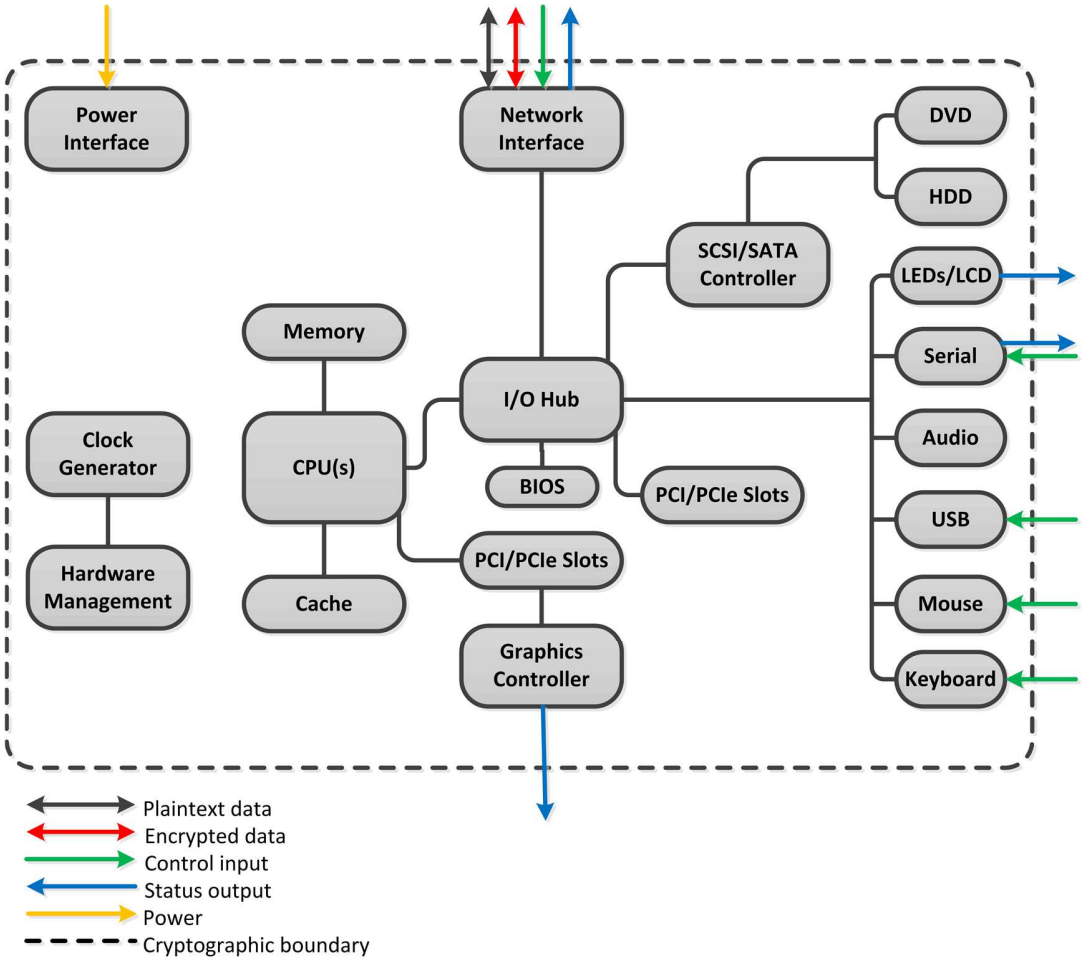

Figure 3 General-purpose computer hardware block diagram

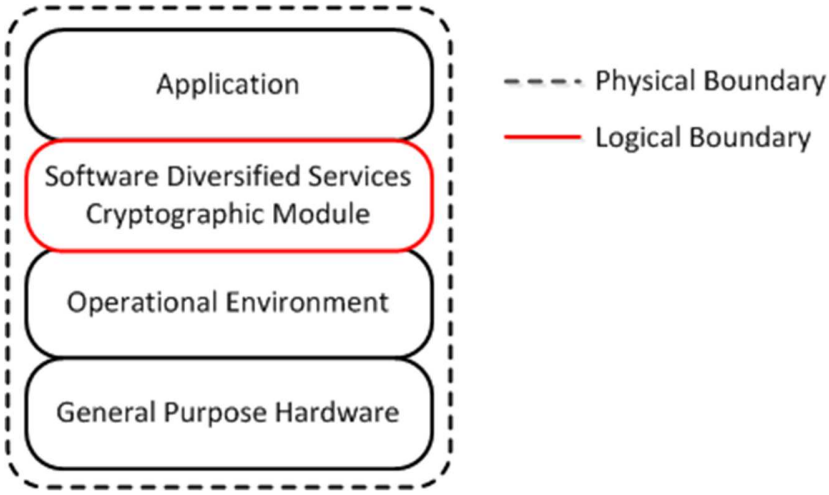

Figure 4 Logical Diagram of the Cryptographic Boundary

2.2.3 Scope of Evaluation

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS 140-2, with Design Assurance at Level 3.

Security Requirements Section	Level
Cryptographic Module Specification	1
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

Figure 5 Security Level specification per individual areas of FIPS 140-2

2.2.4 Cryptographic Algorithms

2.2.4.1 Approved algorithms

The following table provides details of the approved algorithms that are included within the module:

CAVP Cert	Algorithm	Standard	Modes	Key lengths, curves or moduli	Use
#4755	AES	FIPS 197, SP 800-38A	ECB, CBC, CFB8, CFB128, OFB, CTR,	128, 192, 256	Data Encryption-Decryption
#2526	Triple-DES	SP 800-67rev1	TECB, TCBC, TCFB64	192-bits with 168 independent bits providing 112 bits of security strength	Data Encryption-Decryption
#2599	RSA	FIPS 186-2 ANSI X9.31, PKCS#1 v1.5	SHA-224, SHA-256, SHA-384, SHA-512	4096	Signature generation
#2599	RSA	FIPS 186-4 ANSI X9.31, PKCS#1 v1.5	SHA-1[1], SHA-224, SHA-256, SHA-384, SHA-512	1024[2], 2048, 3072	Key generation. Signature generation, Signature verification
Vendor Affirmation	KTS[3] (RSA)	SP 800-56B		2048	Key Transport
#1190	ECDSA	FIPS 186-4	SHA-224,	P-224,	Digital Signature

[1] SHA-1 is only for legacy-use signature verification

[2] 1024-bit key lengths are only for legacy-use signature verification. Other key lengths may be used for key generation and signature generation and signature verification.

[3] KTS (vendor affirmed; key establishment methodology provides 112 bits of encryption strength)

CAVP Cert	Algorithm	Standard	Modes	Key lengths, curves or moduli	Use
			SHA-256, SHA-384, SHA-512	P-256, P-384, P-521	Generation and Verification
#1394	RSADP	FIPS 186-4		2048	RSADP Decryption Primitive Component
#1393	KAS ECC	SP 800-56A		P-224, P-256, P-384, P-521	EC Diffie-Hellman Key Agreement
#1276	DSA	FIPS 186-4	SHA-1[4], SHA-224, SHA-256, SHA-384, SHA-512	1024[5], 2048, 3072	Generation of DSA parameters (P, Q, G), Verification of another implementation's DSA parameters, Key generation. Signature generation, Signature verification.
#3898	SHS[6]	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Hashing
#3167	HMAC	FIPS 198-1	HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512		Message Authentication Code
#1635	DRBG	SP 800-90A	HMAC-SHA-256		Deterministic Random Bit Generation

Figure 6 Approved Algorithms

2.2.4.2 Non-approved algorithms allowed in approved mode

Algorithm	Caveat	Use
NDRNG		Entropy source to seed the random number generator.

Figure 7 Allowed Algorithms

2.2.4.3 Non-approved algorithms

There are no non-approved algorithms included within the module.

[4] SHA-1 only to be used for signature verification and protocol use

[5] 1024-bit key lengths are only to be used for signature verification. Other key lengths may be used for key generation and signature generation and signature verification.

[6] SHA-1 for non-digital signature applications:
SHA-1 is not allowed for digital signature generation. For all other hash function applications, the use of SHA-1 is acceptable. The other applications include HMAC, Key Derivation Functions (KDFs), Random Number Generation (RNGs and RBGs), and hash-only applications (e.g., hashing passwords and using SHA-1 to compute a checksum, such as the approved integrity technique specified in Section 4.6.1 of [FIPS 140-2]).

2.2.5 Components excluded from the security requirements of the standard

There are no components excluded from the security requirements of the standard.

2.3 Physical ports and logical interfaces

The module is classified as a multi-chip standalone module for FIPS 140-2 purposes. The module's physical boundary is that of the general-purpose computer on which it is installed and the physical ports and physical interfaces are those of that GPC. The device shall be running a supported operating system (OS) and supporting all standard interfaces, including keys, buttons and switches, and data ports.

The module provides its logical interfaces via Application Programming Interface (API) calls. This logical interface exposes services (described in section 2.4.2) that the User and operating system utilize directly.

The logical interfaces provided by the module are mapped onto the FIPS 140-2 logical interfaces: data input, data output, control input, and status output as follows:

FIPS 140-2 Logical Interface	Module Mapping
Data Input	Parameters passed to the module via API calls
Data Output	Data returned from the module via API calls
Control Input	API Calls and/or parameters passed to API calls
Status Output	Information received in response to API calls
Power Interface	There is no separate power or maintenance access interface beyond the power interface provided by the GPC itself

Figure 8 Module Interfaces

2.4 Roles, Services and Authentication

2.4.1 Roles

The Cryptographic Module implements both a Crypto Officer role and a User role. Roles are assumed implicitly upon accessing the associated services. Section 2.4.2 summarizes the services available to each role.

Role	Description
Crypto Officer	The administrator of the module having full configuration and key management privileges.
User	General User of the module

Figure 9 Roles

2.4.2 Services

Most of the services provided by the module are provided via access to API calls using interfaces exposed by the module.

However, some of the services, such as power-up module integrity testing are performed automatically and so have no function API, but do provide status output.

Service	Role	Service input	Service output	Description
Generate a key	Crypto Officer	Key Type <RSA, DSA,	Asymmetric key pair	Creates a key object and puts it into the supplied context. By default generates

Service	Role	Service input	Service output	Description
		ECDSA>, User ID, Symmetric algorithm type		a signing key and an encryption sub-key. Assumes "OpenPGP v4". There are options to control the start date, expiration data, type and size of key, the passphrase and the compression algorithm. There is an option to create only the signing key and no sub-keys.
Delete a CSP object	Crypto Officer	CSP ID	Status (Success/Fail)	Deletes a CSP object. Removes the object from memory by first zeroing the context parameters and then freeing the memory used to store the object. This is the service that provides key destruction.
Export a key	Crypto Officer	Key ID	Exported key (file)	Exports a key. Exporting a private key is not allowed.
Import a key	Crypto Officer	Key file	A validated key in a key ring, key ID, status (key, sub-keys, etc.)	Imports a key. Importing a private key is not allowed.
Compute shared key	Crypto Officer	Keys, CSPs	Shared key	Uses EC Diffie-Hellman to compute a shared key.
Sign data	User	Key ID, data	Signature	Sign supplied data.
Verify data	User	Data and signature	Status (Success/Fail)	Verifies the signature on the supplied data.
Encrypt data	User	Data and recipient key IDs	Encrypted message	Encrypts data[7]. Generates a symmetric key[8] to encrypt the data. This is encapsulated separately by each of the recipient keys and appended to the message.
Decrypt data	User	Encrypted message	Data	Decrypts supplied message. Finds the symmetric key in the message encapsulated with module's own recipient key and recovers it and then uses it to decrypt the message.
Hash data	User	Data	Hash of data	Hashes the supplied data.
HMAC data	User	Data	HMAC of data	Calculates an HMAC on the supplied data.
Self-tests	User	N/A	Pass/Fail	Reboot to initiate power-up self-tests. If fail then the module will not run.
Show status	User	N/A	Power-up Self-test passed/failed	Displays the module power-up self-test status.
Get version	N/A	N/A	Version information	Allows the operator to determine if they are using the FIPS version.

[7] IG A.13: The module allows a single Triple-DES encryption key to be used up to a maximum of 16 times. Each time a key is used it can encrypt a maximum of 64K blocks. This a maximum of 2^{20} blocks, which is the limit allowed by this IG.
[8] Module creates a symmetric key that is unmodified output from the DRBG.

Figure 10 Approved Services

2.4.3 Authentication

The module does not support any operator authentication mechanisms. The module does not perform authentication.

2.5 Physical Security

The Cryptographic Module is a software-only cryptographic module and therefore the physical security requirements of FIPS 140-2 do not apply.

2.6 Operational Environment

The Cryptographic Module has been tested on and found to be conformant with the requirements of FIPS 140-2 overall Level 1 on the following GPC platforms:

Operating System	Platform	CPU	AES-NI[9]	SSE2[10]
Windows Server 2012 R2	Dell XPS 8700	Intel i7	X	X
Windows Server 2016	Dell XPS 8700	Intel i7	X	X
Red Hat Server Version 6.9	Dell XPS 8700	Intel i7	X	X
Red Hat Server version 7.4	Dell XPS 8700	Intel i7	X	X
AIX Version 6.1	IBM Power System S822	Power8		
AIX version 7.2	IBM Power System S822	Power8		

Figure 11 Certified Operational Environments

The module is also capable of running on the following platforms but has not been tested during this evaluation and no compliance is being claimed on these platforms:

- Windows 10
- Windows Server 2008 R2
- AIX Version 7.1
- Any Version of Red Hat 6.x
- Any Version of Red Hat 7.x

When porting to an untested platform the following caveat applies: “No assurance of the minimum strength of generated keys”.

The cryptographic module runs in the thread context of the calling application. This provides it with protection from all other processes, preventing access to all keys, intermediate key generation values, and other CSPs.

If an application starts a new instance of the module, then that is a separate instance with its own operational environment. Each new instance only has a single operator (its owner) and the module does not support concurrent operators.

The task scheduler and architecture of the operating system maintain the integrity of the cryptographic module.

[9] AES-NI: The module will make use of the AES-NI instruction set if it is available on the CPU.
[10] SSE2: The SSE2 instructions are used to move the Blocks/Keys efficiently by using the Special SSE2 128-bit registers. The SSE2 instruction are also used in some modes (like CFB/CBC/OFB) to perform XOR using 128bits.

2.7 Cryptographic Key Management

2.7.1 Random Number Generators

The module contains an SP 800-90A approved HMAC DRBG (using SHA-256). Checks are made to ensure that the quality of the entropy remains high enough to be used to seed the DRBG.

Entropy is provided in Windows and Linux by RdRand (an instruction for returning random numbers from an Intel on-chip hardware random number generator which has been seeded by an on-chip entropy source) and in AIX by the PowerPC hardware random number generator accessed via /dev/random. 6,400-bits of entropy are collected for every 256-bits of entropy required. The entropy seeds the DRBG via mechanisms specific to the operational environment in such a way as to guarantee that 256 bits of output from the DRBG has 256 bits of entropy.

2.7.2 Key Generation

The module generates keys using an approved key generation mechanism made up of an SP 800-90A HMAC DRBG and available entropy conditioned by the operational environment.

2.7.3 Key Table

The following tables list all of the keys and CSPs within the module, describe their purpose, and describe how each key is generated, entered and output, stored and destroyed.

Note: “Service” keys. A number of the service APIs are for functions that perform cryptographic operations. Some of these accept keys as parameters. There are also APIs for functions that generate keys and pass them back to the calling application. These keys are ephemeral. They are not stored within the module. After these keys have been used by the API functions, they are zeroized within the module. It is the responsibility of the calling application to ensure that it stores, handles and destroys keys appropriately.

Key	Purpose	Length/ strength	Generation/ establishment	Storage	Entry/ output	Zeroization
Public key[11]	Key encapsulation or signing	DSA: 1024[12], 2048, 3072, or 4096[13] bits. RSA: 1024[14], 2048, 3072, or 4096 bits. ECDSA: P-224, P-256, P-384, P-521 curves.	Generated using the “Generate a key” service.	Not stored within the module.	Public key certificate	Key is zeroized when it is released.
Private key[15]	Key encapsulation	DSA: 1024[16], 2048, 3072, or	As per public key	Not stored within the	N/A	Key is zeroized when it is

[11] Public keys are used to verify digital signatures or to encapsulate keys.
[12] Only for legacy-use signature verification.
[13] 4096-bit modulus is not specified in SP 800-56B and so are not approved, however all modulus sizes of 2048 bits and higher are allowed and.can be used for key encapsulation in an approved mode of operation. 4096 bit keys are allowed for KeyGen, PKCS#1 v1.5 SigGen and for PKCS 1.5 SigVer. 4096-bit keys are generated and used in a way that is compliant with IG A.14.
[14] Only for legacy-use signature verification.
[15] Note private keys are used to create digital signatures and for decryption encapsulated keys.

Key	Purpose	Length/ strength	Generation/ establishment	Storage	Entry/ output	Zeroization
	or signing	4096 bits. RSA: 1024[17], 2048, 3072, or 4096 bits. ECDSA: P-224, P-256, P-384, P-521 curves.		module.		released.
Symmetric key	Data/Message encryption	AES: 128, 192 or 256 bits Triple-DES: 192-bits with 168 independent bits providing 112 bits of security strength	Generated by the "encrypt data" service	Not stored within the module.	Encrypted by asymmetric algorithm (key encapsulation) and appended to encrypted message.	Key is zeroized when it is released.
HMAC key	HMAC	512 bits	Generated using the "Generate a key" service or externally generated.	Not stored within the module.	Service input parameter	Key is zeroized when it is released.
HMAC DRBG "Key" CSP	Internal DRBG variable[18]	512 bits	Initial value of 64 bytes all set to "0x00"	Not stored within the module.	N/A	Key is zeroized when the DRBG is uninstantiated.
HMAC DRBG "V" CSP	Internal DRBG variable[19]	512 bits	Initial value of 64 bytes all set to "0x01"	Not stored within the module.	N/A	Key is zeroized when the DRBG is uninstantiated.
HMAC DRBG seed and entropy CSPs	Internal DRBG variable[20]	6400 bits	non-approved NDRNG	Not stored within the module.	N/A	Key is zeroized when the DRBG is uninstantiated.
KAS ECC Public Key	Key agreement	P-224, P-256, P-384, P-521 curves.	Generated using the "Generate a key" service.	Not stored within the module.	Public key certificate	Key is zeroized when it is released.
KAS ECC	Key	P.224, P.256,	As per public	Not stored	N/A	Key is zeroized

[16] Only for legacy-use signature verification.
[17] Only for legacy-use signature verification.
[18] These are variables used internally by the HMAC DRBG that are required by Implementation Guidance 14.5 to be listed in the Cryptographic Module Security Policy document. There is an initial seed, and the algorithm is reseeded from a non-approved NDRNG.
[19] See above footnote.
[20] See above footnote.

Key	Purpose	Length/ strength	Generation/ establishment	Storage	Entry/ output	Zeroization
Private Key	agreement	P.384, P.521 curves.	key	within the module.		when it is released.
Shared Key	Operator-defined	Operator-defined	"Compute shared key" service	Not stored within the module.	Key is "shared" between two parties during key establishment	Key is zeroized when it is released.

Figure 12 Key Table

2.7.4 Key Destruction

All key material managed by the module can be zeroized using the "Delete a CSP Object" service.

In this way all key material and CSPs are zeroized. There are no user-accessible plaintext keys or CSPs in the module.

2.7.5 Access to Key Material

The following table shows the access that an operator has to specific keys or other critical security parameters when performing each of the services relevant to his/her role.

Service	Role	Public key	Private key	Symmetric key	HMAC key	HMAC DRBG key	HMAC DRBG V	HMAC DRBG seed and entropy CSPs	KAS ECC Public Key	KAS ECC Private Key	Shared Key
Generate a key	Crypto Officer	W	W			U	U	U	W	W	
Delete a key	Crypto Officer	W	W	W	W	W	W	W	W	W	W
Export a key	Crypto Officer	R									
Import a key	Crypto Officer	W									
Compute shared key	Crypto Officer	U	U						U	U	W
Sign data	User		U								
Verify data	User	U									
Encrypt data	User		U	U		U	U	U			U
Decrypt data	User		U	U							U
Hash data	User										
HMAC data	User				U						
Self-tests	N/A										

Service	Role	Public key	Private key	Symmetric key	HMAC key	HMAC DRBG key	HMAC DRBG V	HMAC DRBG seed and entropy CSPs	KAS ECC Public Key	KAS ECC Private Key	Shared Key
Show status	N/A										
Get version	N/A										

Figure 13 Access to keys by services

Access Rights	Blank	N/A
	R	Read
	W	Write
	U	Use

Note: Key zeroization zeroes all keys and CSPs, this is a “write” operation in that all keys are overwritten with zeroes.

2.8 Self-Tests

The module implements both power-up and conditional self-tests as required by FIPS 140-2.

The following two sections outline the tests that are performed.

2.8.1 Power-up self-tests

After power-cycling or booting the appliance the module executes the Power-Up Self-Tests with no further inputs or actions by the operator. The module meets the requirements of IG 9.10 regarding power-up self-tests for software module libraries.

The module implements the following power-up self-tests. The module inhibits all data output while it is operating in the Self-Test state.

Object	Test
AES	AES-128-ECB encrypt known answer test AES-128-ECB decrypt known answer test AES-192-ECB encrypt known answer test AES-192-ECB decrypt Known answer test AES-256-ECB encrypt known answer test AES-256-ECB decrypt known answer test AES-128-CBC encrypt known answer test AES-128-CBC decrypt known answer test AES-192-CBC encrypt known answer test AES-192-CBC decrypt Known answer test AES-256-CBC encrypt known answer test AES-256-CBC decrypt known answer test AES-128-CFB8 encrypt known answer test AES-128-CFB8 decrypt known answer test AES-192-CFB8 encrypt known answer test

Object	Test
	AES-192-CFB8 decrypt Known answer test AES-256-CFB8 encrypt known answer test AES-256-CFB8 decrypt known answer test AES-128-CFB128 encrypt known answer test AES-128-CFB128 decrypt known answer test AES-192-CFB128 encrypt known answer test AES-192-CFB128 decrypt Known answer test AES-256-CFB128 encrypt known answer test AES-256-CFB128 decrypt known answer test AES-128-OFB encrypt known answer test AES-128-OFB decrypt known answer test AES-192-OFB encrypt known answer test AES-192-OFB decrypt Known answer test AES-256-OFB encrypt known answer test AES-256-OFB decrypt known answer test AES-128-CTR encrypt known answer test AES-128-CTR decrypt known answer test AES-192-CTR encrypt known answer test AES-192-CTR decrypt Known answer test AES-256-CTR encrypt known answer test AES-256-CTR decrypt known answer test
Triple-DES	ECB encrypt known answer test ECB decrypt known answer test CBC encrypt known answer test CBC decrypt known answer test CFB64 encrypt known answer test CFB64 decrypt known answer test
RSA	Known answer test (signature generation) Known answer test (signature verification) Known answer test (encryption) Known answer test (decryption)
DSA	Known answer test (signature generation) Known answer test (signature verification)
ECDSA	Known answer test (signature generation) Known answer test (signature verification)
KAS ECC	Primitive Z computation known answer test
SHA-1	Known answer test
SHA-224	Known answer test
SHA-256	Known answer test
SHA-384	Known answer test
SHA-512	Known answer test
HMAC-SHA-224	Known answer test
HMAC-SHA-256	Known answer test
HMAC-SHA-384	Known answer test
HMAC-SHA-512	Known answer test
DRBG	SP 800-90A HMAC DRBG known answer test SP 800-90A Section 11.3 Health Tests (instantiate, reseed and generate)

Object	Test
Module integrity	RSA 2048 bit digital signature with SHA-256

Figure 14 Power-up self-tests

If any of the power-up KATs fail, the module exits with an error. While in the error state the module inhibits all data output and all cryptographic operations are prohibited. The operator may restart the module to re-run the power up self-tests.

2.8.2 Conditional self-tests

The module implements the following conditional self-tests:

Event	Test	Consequence of Failure
Module requests a random number from the FIPS Approved SP800-90A DRBG	A continuous random number generator test	Module tries again to request a random number.
Module requests a random number from the NDRNG used to seed the FIPS Approved SP800-90A DRBG	A continuous random number generator test	Module tries again to request a random number.
RSA key pair is generated	RSA pair-wise consistency test (sign-verify and encrypt-decrypt)	Key is discarded and module returns an error.
DSA key pair is generated	DSA pair-wise consistency test	Key is discarded and module returns an error.
ECDSA key pair is generated	ECDSA pair-wise consistency test	Key is discarded and module returns an error.

Figure 15 Conditional self-tests

2.9 Design Assurance

SDS employ industry standard best practices in the design, development, production and maintenance of all of its products, including the FIPS 140-2 module.

This includes the use of an industry standard configuration management system that is operated in accordance with the requirements of FIPS 140-2, such that each configuration item that forms part of the module is stored with a label corresponding to the version of the module and that the module and all of its associated documentation can be regenerated from the configuration management system with reference to the relevant version number.

Design documentation for the module is maintained to provide clear and consistent information within the document hierarchy to enable transparent traceability between corresponding areas throughout the document hierarchy, for instance, between elements of this Cryptographic Module Security Policy (CMSP) and the design documentation.

Guidance appropriate to an operator's Role is provided with the module and provides all of the necessary assistance to enable the secure operation of the module by an operator, including the Approved security functions of the module.

Delivery of the Cryptographic Module to customers from the vendor is via secure download. The module firmware downloaded can be verified using SHA-256 hash values that are downloaded separately.

2.10 Mitigation of Other Attacks

The module does not mitigate any other attacks.

3 Secure Operation

The module only operates in a secure mode of operation and there are no specific installation steps that need to be taken. The module binary can simply be placed in a folder on a GPC disk drive and accessed by a calling application. The module does not have a non-approved mode of operation.