
Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program

**National Institute of Standards and Technology
Canadian Centre for Cyber Security**



Initial Release: September 21, 2020

Last Update: September 21, 2020

Table of Contents

OVERVIEW	4
SECTION 1 – GENERAL	5
SECTION 2 – CRYPTOGRAPHIC MODULE SPECIFICATION	6
2.3.A BINDING OF CRYPTOGRAPHIC ALGORITHM VALIDATION CERTIFICATES.....	6
2.3.B SUB-CHIP CRYPTOGRAPHIC SUBSYSTEMS	7
2.3.C PROCESSOR ALGORITHM ACCELERATORS (PAA) AND PROCESSOR ALGORITHM IMPLEMENTATION (PAI)	11
2.4.A DEFINITION AND USE OF A NON-APPROVED SECURITY FUNCTION	13
2.4.B TRACKING THE COMPONENT VALIDATION LIST	17
SECTION 3 – CRYPTOGRAPHIC MODULE INTERFACES	19
3.4.A TRUSTED CHANNEL	19
SECTION 4 – ROLES, SERVICES, AND AUTHENTICATION	22
4.1.A AUTHORISED ROLES	22
4.4.A MULTI-OPERATOR AUTHENTICATION	23
SECTION 5 – SOFTWARE/FIRMWARE SECURITY	26
5.A NON-RECONFIGURABLE MEMORY INTEGRITY TEST.....	26
SECTION 6 – OPERATIONAL ENVIRONMENT	27
SECTION 7 – PHYSICAL SECURITY	28
7.3.A TESTING TAMPER EVIDENT SEALS	28
7.3.B HARD COATING TEST METHODS (LEVEL 3 AND 4)	29
SECTION 8 – NON-INVASIVE SECURITY	31
SECTION 9 – SENSITIVE SECURITY PARAMETER MANAGEMENT	32
9.3.A ENTROPY CAVEATS	32
9.5.A SSP ESTABLISHMENT AND SSP ENTRY AND OUTPUT.....	35
9.6.A ACCEPTABLE ALGORITHMS FOR PROTECTING STORED SSPs	42
9.7.A ZEROIZATION OF ONE TIME PROGRAMMABLE (OTP) MEMORY	44
SECTION 10 – SELF-TESTS	45
10.3.A CRYPTOGRAPHIC ALGORITHM SELF-TEST REQUIREMENTS	45
10.3.B SELF-TEST FOR EMBEDDED CRYPTOGRAPHIC ALGORITHMS.....	51
SECTION 11 – LIFE-CYCLE ASSURANCE	53
SECTION 12 – MITIGATION OF OTHER ATTACKS	54
ANNEX A – DOCUMENTATION REQUIREMENTS.....	55
ANNEX B – CRYPTOGRAPHIC MODULE SECURITY POLICY	56
ANNEX C – APPROVED SECURITY FUNCTIONS	57
C.A USE OF NON-APPROVED ELLIPTIC CURVES.....	57
C.B VALIDATION TESTING OF HASH ALGORITHMS AND HIGHER CRYPTOGRAPHIC ALGORITHM USING HASH ALGORITHMS	58
C.C THE USE AND THE TESTING REQUIREMENTS FOR THE FAMILY OF FUNCTIONS DEFINED IN FIPS 202.....	59
C.D USE OF A TRUNCATED HMAC	60
C.E KEY GENERATION FOR RSA SIGNATURE ALGORITHM	62
C.F APPROVED MODULUS SIZES FOR RSA DIGITAL SIGNATURE FOR FIPS 186-4	62
C.G SP 800-67REV2 LIMIT ON THE NUMBER OF ENCRYPTIONS WITH THE SAME TRIPLE-DES KEY.....	64

C.H KEY/IV PAIR UNIQUENESS REQUIREMENTS FROM SP 800-38D.....	66
C.I XTS-AES KEY GENERATION REQUIREMENTS	73
C.J REQUIREMENTS FOR TESTING TO SP 800-38G	74
ANNEX D – APPROVED SENSITIVE SECURITY PARAMETER GENERATION AND ESTABLISHMENT METHODS	76
D.A ACCEPTABLE SSP ESTABLISHMENT PROTOCOLS.....	76
D.B STRENGTH OF SSP ESTABLISHMENT METHODS	77
D.C REFERENCES TO THE SUPPORT OF INDUSTRY PROTOCOLS	80
D.D ELLIPTIC CURVES AND THE FFC SAFE-PRIME GROUPS IN SUPPORT OF INDUSTRY PROTOCOLS	81
D.E ASSURANCE OF THE VALIDITY OF A PUBLIC KEY FOR SSP ESTABLISHMENT	83
D.F KEY AGREEMENT METHODS	84
D.G KEY TRANSPORT METHODS.....	87
D.H REQUIREMENTS FOR VENDOR AFFIRMATION TO SP 800-133	91
D.I THE USE OF POST-PROCESSING IN KEY GENERATION METHODS	92
D.J ENTROPY ESTIMATION AND COMPLIANCE WITH SP 800-90B	95
D.K INTERPRETATION OF SP 800-90B REQUIREMENTS	96
D.L CRITICAL SECURITY PARAMETERS FOR THE SP 800-90A DRBGs.....	102
D.M USING THE SP 800-108 KDFs IN AN APPROVED MODE	103
D.N SP 800-132 PASSWORD-BASED KEY DERIVATION FOR STORAGE APPLICATIONS.....	104
ANNEX E – APPROVED AUTHENTICATION MECHANISMS	106
ANNEX F – APPROVED NON-INVASIVE ATTACK MITIGATION TEST METRICS.....	107
CHANGE SUMMARY	108
NEW GUIDANCE	108
MODIFIED GUIDANCE	108
MAPPING IGS OF FIPS 140-3 TO FIPS 140-2	108
END OF DOCUMENT	111

Overview

This Implementation Guidance document is issued and maintained by the U.S. Government's National Institute of Standards and Technology ([NIST](#)) and the Canadian Centre for Cyber Security ([CCCS](#)), which serve as the validation authorities of the Cryptographic Module Validation Program ([CMVP](#)) for their respective governments. The CMVP validates the test results of National Voluntary Laboratory Accreditation Program ([NVLAP](#)) accredited Cryptographic and Security Testing ([CST](#)) Laboratories which test cryptographic modules for conformance to Federal Information Processing Standard Publication (FIPS) 140-3, [Security Requirements for Cryptographic Modules](#). The Cryptographic Algorithm Validation Program ([CAVP](#)) addresses the testing of [Approved Security Functions](#) and [Approved Sensitive Security Parameter Generation and Establishment Methods](#) which are referenced in the SP 800-140 series of FIPS 140-3.

This document is intended to provide programmatic guidance of the CMVP, and in particular, clarifications and guidance pertaining to ISO/IEC 24759:2017(E), *Test requirements for cryptographic modules*, which are further clarified in [FIPS PUB 140-3 Derived Test Requirements \(DTR\)](#), which are used by CST Laboratories to test for a cryptographic module's conformance to FIPS 140-3. Guidance presented in this document is based on responses issued by NIST and CCCS to questions posed by the CST Labs, vendors, and other interested parties. *Information in this document is subject to change by NIST and CCCS.*

Each section of this document corresponds with a requirements section of ISO/IEC 19790:2012 (corrections made in 2015). Within each section, the guidance is listed according to a subject phrase. For those subjects that may be applicable to multiple requirements areas, they are listed in the area that seems most appropriate. Under each subject there is a list, including the date of issue for that guidance, along relevant assertions, test requirements, and vendor requirements from the DTR. (*Note: For each subject, there may be additional test and vendor requirements which apply.*) Next, there is section containing a question or statement of a problem, along with a resolution and any additional comments with related information. This is the implementation guidance for the listed subject.

Cryptographic modules validation listings can be found at:

- [Cryptographic Module Validation Lists](#)

Cryptographic algorithm validation listings can be found at:

- [Cryptographic Algorithm Validation Lists](#)
-

Section 1 – General

Section 2 – Cryptographic module specification

2.3.A Binding of Cryptographic Algorithm Validation Certificates

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

Cryptographic algorithm implementations are tested and validated under the Cryptographic Algorithm Validation Program (CAVP). The cryptographic algorithm validation certificate states the name and version number of the validated algorithm implementation, and the tested operational environment.

Cryptographic modules are tested and validated under the Cryptographic Module Validation Program (CMVP). The cryptographic module validation certificate states the name and version number of the validated cryptographic module, and the tested operational environment.

The validation certificate serves as a benchmark for the configuration and operational environment used during the validation testing.

Question/Problem

What are the configuration control and operational environment requirements for the cryptographic algorithm implementation(s) embedded within a cryptographic module when the latter is undergoing testing for compliance to FIPS 140-3?

Resolution

For a validated cryptographic algorithm implementation to be embedded within a software, firmware or hardware cryptographic module that undergoes testing for compliance to FIPS 140-3, the following requirements must be met:

1. The implementation of the validated cryptographic algorithm has not been modified upon integration into the cryptographic module undergoing testing; and
2. The operational environment under which the validated cryptographic algorithm implementation was tested by the CAVP must be identical to the operational environment that the cryptographic module is being tested under by the CST laboratory, subject to the following rules:
 - For software modules, each Operational Environment listed must consist of the Operating System, the platform, and the processor on which the module was tested. If a hypervisor was used, that must also be listed (see the [Management Manual - Annex A](#))
 - If an implementation has been tested on an X-bit processor (e.g. 32-bit, 64-bit), a claim cannot be made that the implementation also runs on different bit size processors.

For example: An algorithm implementation was tested and validated on a 32-bit platform. This was used in a previous 32-bit version of a software module that was validated for conformance to FIPS 140-3. Now the software module is undergoing testing on a 64-bit platform. This software module cannot operate on a 32-bit platform without change. In this case the operational

environments are not the same; therefore, the algorithm implementations must be re-tested on the 64-bit platform. Memory size, processor frequency, etc. are not relevant.

- If an algorithm implementation has been tested on one operating system, a claim cannot be made that the implementation also runs on another operating system when it is considered for module testing.

The algorithm implementation must have been tested on every operating environment claimed by the software module. The algorithm certificate may include other operating environments as well, but they are not relevant to the module under test.

- If algorithm testing is not performed directly by the CST Lab, the CST Lab is responsible for asking the vendor to supply the operating environment (processor and/or operating system and platform) on which they ran the algorithm implementation and with which they generated the vector set test results. It is the CST Labs' responsibility to verify that the results in the vector set test results were generated using the specified operating environment.
- If an algorithm is implemented in HDL on a Field Programmable Gate Array (FPGA) device and there is no underlying "OS" implemented in the FPGA, the algorithm implementation cannot be validated as firmware and ported as is to other FPGAs, since the CMVP does not validate HDL (which is equivalent to source code). The algorithm implementation would be validated in the FPGA as hardware.

Once the FPGA device is validated, one could take the HDL on this FPGA and reuse it in creating a new FPGA. If this were done, the algorithm implementations would need to be validated on the new hardware because they would be considered as new hardware implementations.

Additional Comments

Additional information regarding operational environment can be found in the [CAVP FAQ](#) GEN.12.

2.3.B Sub-Chip Cryptographic Subsystems

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	
Relevant Test Requirements:	
Relevant Vendor Requirements:	

Background

Increased levels of integration in IC design, such as ASIC, FPGA or System-on-Chip (SoC), have been developed with heterogeneous computing characteristics. Heterogeneous computing may include multiple processors or functional engines, with isolated security subsystem designs that may be re-used in multiple configurations or generations of products.

Question/Problem

What is a *sub-chip cryptographic subsystem*, and what are the requirements for *initial* validation? Once validated, how can the sub-chip cryptographic subsystem be re-validated if modified? How can a non-modified sub-chip cryptographic subsystem be ported and reused on other single-chip implementations?

Resolution

The following terminology is used in the context of this IG:

HDL – Hardware Design Language; examples are Verilog and VHDL.

Security relevant – relevant to the requirements of FIPS 140-3.

Soft circuitry core – an uncompiled hardware subsystem of an ASIC, FPGA or SoC.

Hard circuitry core – a fixed or precompiled hardware subsystem of an ASIC, FPGA or SoC.

For a hardware module, the minimum defined physical boundary in **ISO/IEC 19790** is a single-chip. For single-chip hardware modules a sub-chip cryptographic subsystem may be defined as the set of hard and/or soft circuitry cores and associated firmware which represents a sub-chip cryptographic subsystem boundary of a single-chip hardware module. The sub-chip cryptographic subsystem is integrated on the single-chip which may contain other functional subsystems (e.g. processor(s), memory, I/O and internal bus controls, sensors, etc.) and associated firmware. Upon fabrication of the complete physical single-chip, the HDL will be transformed to a gate or physical circuitry representation which may or may not retain a definable internal sub-chip cryptographic subsystem boundary.

1. Initial validation or security relevant re-validations

- The physical boundary **shall** be defined as the single-chip physical boundary;
 - **ISO/IEC 19790:2012** Section 7.7 requirements **shall** apply at the physical boundary
- **ISO/IEC 19790** defines the Cryptographic boundary as an explicitly defined perimeter that establishes the boundary of all components (i.e. set of hardware, software or firmware components) of the cryptographic module. For a sub-chip cryptographic subsystem, the physical boundary is the single-chip physical boundary while its Hardware Module Interface (HMI) (i.e. the sub-chip cryptographic subsystem boundary) is defined as the set of hard and/or soft circuitry cores and associated firmware that comprises the sub-chip cryptographic subsystem.
- If there is any associated firmware externally loaded into the sub-chip cryptographic subsystem, the associated firmware **shall** meet requirements of Software/Firmware Load Test (**ISO/IEC 19790:2012** Section 7.10.3.4).
- Except for externally loaded firmware, the associated firmware **shall** be stored and loaded inside the sub-chip cryptographic subsystem and **shall** meet the pre-operational software/firmware integrity test (**ISO/IEC 19790:2012** Section 7.10.2.2).
- The ports and interfaces (**ISO/IEC 19790:2012** Section 7.3) **shall** be defined at the HMI.
 - For operational testing purposes, access to the HMI ports **shall** be required and a mapping **shall** be provided. These may be mapped to physical I/O pins, internal test interfaces (e.g. Level Sensitive Scan Design (LSSD)) or the HMI data and control ports. The tester **shall** demonstrate that the ports at the HMI are accessible via the single-chip's other functional subsystems in a manner such that following five kinds of information are provably unmodifiable and under control of the test program:
 - Data input,
 - Data output,
 - Control input,
 - Control output, and
 - Status output

even in the presence of intervening other functional subsystems.

Note 1: Typically, the test program acting on behalf of the tester with direct access to the ports and interfaces defined at the HMI provides the required demonstration of port access.

Note 2: In single-chip embodiments, there may be intervening functional subsystems (or intervening circuitry) other than the sub-chip cryptographic subsystem subject to

testing. There is a security concern that such intervening subsystems might act maliciously (e.g. intercept, modify, and store CSPs, or attempt a replay attack and/or man-in-the-middle attack). The tester **shall** verify and provide the vendor's rationale in the validation report (TE02.13.01) explaining existing risks and mitigations. The CMVP may provide additional guidance in the future on how to analyze and document such potential security risks.

Note 3: If applicable, VE04.51.01 and TE04.51.01 **shall** be considered at the level of the tested sub-chip cryptographic subsystem and potential differences between the internal and external with respect to the subsystem boundary single chip clocks **shall** be accounted for properly.

- Depending on the security level, the requirements for sensitive security parameter establishment (ISO/IEC 19790:2012 Section 7.9.4-5) **shall** be applicable at the HMI.
 - Transferring SSPs including the entropy input between a sub-chip cryptographic subsystem and an intervening functional subsystem for Levels 1 and 2 on the same single chip is considered as not having Sensitive Security Parameter Establishment crossing the HMI of the sub-chip module per [IG 9.5.A](#). Nevertheless, the above Note 2 for the ports and interfaces is applicable for the transferring of SSPs as well. That is, the tester **shall** provide a rationale in the physical security test report explaining risks and mitigations to the malicious act by such intervening subsystems.
 - For Security Levels 3 and 4 modules, SSP establishment is ED / EE as stated in [IG 9.5.A](#).
 - Versioning information (AS04.13) **shall** be provided for the
 - physical single-chip including any excluded functional subsystem firmware (**shall** be specified in the OE field of the validation),
 - the sub-chip cryptographic subsystem soft and hard circuitry cores, and
 - the associated firmware.
 - Processor sub-functions outside the HMI but within the physical boundary such as a processor, memory macros, I/O controllers, etc. **may** be excluded under AS02.13 and AS02.14. However, the data paths used to meet either AS03.18 and AS03.20-22 or AS03.19 and AS03.20-22 (**depending on the level**) **shall not** be excluded.
2. **Non-security relevant re-validations associated with changes within physical boundary**
Existing re-validation guidance is applicable.
3. **Sub-chip cryptographic subsystem porting**
The sub-chip cryptographic subsystem may be ported to other single-chip implementations which may be different chip technologies, and/or different non-security relevant functional subsystems.

A sub-chip cryptographic subsystem that was previously validated in a single-chip can be ported to other single-chip constructs as a 3MU/3MC submission to the CMVP. The following is applicable to validate this new single-chip module as a 3MU/3MC:

- The laboratory **shall** verify that there are no security relevant changes in the sub-chip cryptographic subsystem;
- If an entropy source is contained within the sub-chip cryptographic subsystem, a new entropy estimate **shall** be provided;

Note 1: A new entropy estimate may not be required, if the entropy is collected outside the sub-chip cryptographic subsystem, depending on changes to the entropy source or the subsystem housing it. Please refer to [IG 9.3.A](#) and [IG D.J](#) for details on applicable caveats and entropy estimates.

Note 2: Single chip embodiments may implement an ENT or a DRBG linked to a dedicated entropy source (ENT) inside the physical boundary. Such cases may be implemented (a) inside the sub-

chip cryptographic subsystem or (b) in two or more sub-chip cryptographic subsystems. The case (b) represents multiple disjoint sub-chip cryptographic subsystems (see 4 of this IG).

- Approved security functions **shall** be retested and validated by the CAVP if implemented in a soft circuitry core recompiled in a different part configuration.

Note 3: If the original algorithm testing was performed as stated in the Management Manual Section 7.2 – *Testing using Emulators and Simulators* in a module simulator, and there is no change to the soft-core, no additional algorithm testing is required.

- Operational regression testing ([Management Manual \[IG G.8.1 Table\]](#)) **shall** be performed on the new sub-chip cryptographic subsystem after fabrication (transformation of the HDL to a gate or physical circuitry representation);
- **ISO/IEC 19790:2012** Section 7.3 **shall** be addressed for the new single-chip module for all Security Levels within this Section.
- **ISO/IEC 19790:2012** Section 7.7 **shall** be addressed for the new single-chip module at Security Level 1.
- **ISO/IEC 19790:2012** Sections 7.11.2 and 7.11.9 **shall** be addressed for the new single-chip module for all Security Levels within this Section.
- A new Security Policy **shall** be provided for the new single-chip module.
- A new validation certificate will be issued. Versioning information **shall** be provided for
 - the new physical single-chip
 - non-security relevant single-chip functional subsystem firmware if applicable,
 - the sub-chip cryptographic subsystem soft and hard circuitry cores (which are unchanged from the original validation), and
 - the associated firmware.

The testing laboratory **shall** submit a 3MU/3MC test report for the ported updated sub-chip cryptographic subsystem to the CMVP. NIST Cost Recovery fee is applicable.

4. Multiple disjoint sub-chip cryptographic subsystems:

Disjoint sub-chip cryptographic subsystems may exist on a single-chip. Each **shall** be separately validated.

Transferring Keys/SSPs including the entropy input between two disjoint sub-chip cryptographic subsystems on the same single chip for Security Level 1 or Security Level 2 modules is considered not having SSP establishment across their sub-chip cryptographic subsystem boundary per [IG 9.5.A](#).

- For Level 3 and Level 4 modules, CSP establishment is ED / EE as stated in [IG 9.5.A](#).

Alternatively, plaintext CSPs may be shared directly between two disjoint sub-chip cryptographic subsystems via a Trusted Channel (**ISO/IEC 19790:2012** Section 7.3.4). In this scenario, the following porting rules **shall** apply:

- a. If the two sub-chip modules that are connected by a Trusted Channel are ported together, it is considered security relevant and the testing lab **shall** submit a 3MU/3MC or a 5FS.
- b. If only one of the sub-chip modules that are connected by a Trusted Channel is ported, then the testing lab **shall** verify that the Trusted Channel is no longer functional and may submit a 3MU/3MC.
- c. If only one of the sub-chip modules that are connected by a Trusted Channel are ported and it is connected to a new sub-chip module, then it is considered security relevant and the testing lab **shall** submit a 3MU/3MC or a 5FS.

Additional Comments

This IG does not apply to single-chip implementations that do not contain sub-chip cryptographic subsystems, i.e. there is only one boundary which is the physical boundary.

If the sub-chip cryptographic subsystem enters an error state, the FIPS 140-3 requirements are applicable at the HMI of the sub-chip cryptographic subsystem; not at the boundary of the single-chip.

2.3.C Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI)

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	
Relevant Test Requirements:	
Relevant Vendor Requirements:	

Background

Single-chip processor manufacturers are adding acceleration functions to support complex cryptographic algorithms. When these functions are added, the CMVP, the CAVP and the Cryptographic Technology group at NIST will determine if the acceleration function is simply a mathematical construct or a complete cryptographic algorithm as defined in the NIST standards.

If the function is deemed the complete cryptographic algorithm, then FIPS 140-3 defines the component to be security-specific hardware. Complete documentation of the entire component, including HDL, **shall** be submitted to the testing laboratory when under test. This type of implementation is considered a Processor Algorithm Implementation (PAI) function. If the module has been designed to run with and without the security-specific hardware, the resolution below under Software/Firmware Module may apply.

If the function is deemed a mathematical construct and not the complete cryptographic algorithm as defined in the NIST standards, then FIPS 140-3 does not define the component to be security-specific hardware and complete documentation of the entire component, including HDL, is not required. This type of implementation is considered a Processor Algorithm Acceleration (PAA) function.

Question/Problem

What are the currently known processor chips that include Processor Algorithm Acceleration (PAA) and Processor Algorithm Implementation (PAI) functions to support complex cryptographic algorithms and how is it indicated on the validation certificate?

Resolution

If a cryptographic module is designed to utilize a processor chip that includes PAA and/or PAI, the part number or version of the processor chip **shall** be included in TE02.15.01. A module that utilizes such processor hardware may or may not be defined as a hybrid module.

Software/Firmware-Hybrid Module: If the software or firmware component of the hybrid can only support a cryptographic algorithm by utilizing the PAA or PAI capability, then the module **shall** be defined as a Software/Firmware-Hybrid Module Embodiment.

PAA

- **Module versioning** information **shall** include the part number or version of the processor chip.
- **Operational Environment:** Tested as meeting Level 1 with <OS> running on <platform> with PAA

PAI

- **Module versioning** information **shall** include the part number or version of the processor chip.

- **Operational Environment:** Tested as meeting Level 1 with <OS> running on <platform> with PAI

Software/Firmware Module: If the software or firmware component of the module can support a cryptographic algorithm natively (within the software/firmware) or by utilizing an available PAA or PAI, the module **shall** be defined as a Software/Firmware module Embodiment, unless other requirements designate the module as hybrid.

PAA

- **Algorithm certificates;** the accelerated algorithms **shall** be tested in both software/firmware only execution and PAA execution.
- **Operational Environment:** Tested as meeting Level 1 with <OS> running on <platform> with PAA; <OS> running on <platform> without PAA

PAI

- **Algorithm certificates;** the algorithms **shall** be tested in both software/firmware only execution and PAI execution.
- **Operational Environment:** Tested as meeting Level 1 with <OS> running on <platform> with PAI; <OS> running on <platform> without PAI

Known PAAs:

- Intel Processors – Xeon, Core i5, Core i7, Core M and Atom with Westmere, Sandy Bridge, Ivy Bridge, Haswell, Broadwell, Skylake, Kaby Lake, Coffee Lake, Goldmont Plus, Whiskey Lake, Amber Lake, Cascade Lake, Comet Lake or Sunny Cove micro-architectures: PAA = AES-NI
 - Accelerator sub-functions for AES implementations
- Intel Processors – Atom, Celeron, and Pentium with Goldmont, Goldmont Plus, Sunny Cove micro-architectures: PAA = Intel SHA Extensions
 - Accelerator sub-functions for SHA implementations
- AMD Processors - Opteron, Athlon, Sempron, FX, and A series with Bulldozer, Piledriver, Steamroller, Jaguar, Puma micro-architectures: PAA = AES-NI
 - Accelerator sub-functions for AES implementations
- AMD Processors – Ryzen series with Zen micro-architectures: PAA = SHA Extensions
 - Accelerator sub-functions for SHA implementations
- ARM Cortex A series, R series, Qualcomm Snapdragon, Apple A series processors, Samsung Exynos with ARMv7-A and ARMv8-A micro-architectures: PAA = NEON or Cryptography Extensions
 - Accelerator sub-functions for AES and SHA implementations
- IBM Power Processors 8, 9: PAA = Power ISA
 - Accelerator sub-functions for AES and SHA implementations
- Oracle: Oracle SPARC T series, M series: PAA = SPARC
 - Accelerator sub-functions for AES, DES, and SHA implementations

Known PAIs:

- IBM CP Assist for Cryptographic Functions (CPACF)
 - Full implementations of AES (ECB, CBC), SHA

Additional Comments

1. AES.2 in the CAVP FAQ gives requirements for both types of implementations.
2. The processor manufacturer may provide a device driver to support use of the processor algorithm accelerator. The device driver **shall** not provide any additional functionality to the PAA.
3. The implementation of complete algorithms, partial cryptographic modules, or full cryptographic modules as a component of a single-chip, or multiple of any of the above as components of a single-chip, is addressed in the Sub-Chip Cryptographic Subsystems IG.
4. Please contact the CMVP to address new PAA or PAI implementations to make a determination whether they are full cryptographic functions or not.
5. If the PAI security function appears on the list of known PAIs, its HDL is not required for validation of software modules using it.

2.4.A Definition and Use of a non-Approved Security Function

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.21</i>
Relevant Test Requirements:	<i>TE02.21.01 and TE02.21.02</i>
Relevant Vendor Requirements:	<i>VE02.21.01 and VE02.21.02</i>

Background

ISO/IEC 19790:2012 Terms and Definitions:

Approved mode of operation: set of services which includes at least one service that utilises an approved security function or process and can include non-security relevant services. NOTE 1: Not to be confused with a specific mode of an approved security function, e.g. Cipher Block Chaining (CBC) mode. NOTE 2: Non-approved security functions or processes are excluded.

Approved security function: security function (e.g. cryptographic algorithm) that is referenced in Annex C [which is superseded by **SP 800-140C**. Also see **SP 800-140D** which includes approved security functions in relation to SSP generation and establishment methods].

Cryptographic algorithm: well-defined computational procedure that takes variable inputs, which may include cryptographic keys, and produces an output.

Plaintext key: unencrypted cryptographic key or a cryptographic key obfuscated by non-approved methods which is considered unprotected.

Security function: cryptographic algorithms together with modes of operation, such as block ciphers, stream ciphers, symmetric or asymmetric key algorithms, message authentication codes, hash functions, or other security functions, random bit generators, entity authentication and SSP generation and establishment all approved either by ISO/IEC or an approval authority. NOTE: See Annex C [which is superseded by **SP 800-140C**. Also see **SP 800-140D**].

ISO/IEC 19790:2012 Section 6 Functional Security Objectives:

The security requirements specified in this International Standard relate to the secure design and implementation of a cryptographic module. The security requirements start with a baseline level of security objectives with increasing levels of security objectives. The requirements are derived from the following high-level functional security objectives for a cryptographic module to:

- employ and correctly implement the approved security functions for the protection of sensitive information;

- protect a cryptographic module from unauthorized operation or use;
- prevent the unauthorized disclosure of the contents of the cryptographic module, including CSPs;
- prevent the unauthorized and undetected modification of the cryptographic module and cryptographic algorithms, including the unauthorized modification, substitution, insertion, and deletion of SSPs;
- provide indications of the operational state of the cryptographic module;
- ensure that the cryptographic module performs properly when operating in an approved mode of operation;
- detect errors in the operation of the module and to prevent the compromise of SSPs resulting from these errors;
- ensure the proper design, distribution and implementation of the cryptographic module.

ISO/IEC 19790:2012 Section 7.9.1 Sensitive security parameter management general requirements:

Encrypted CSPs refer to CSPs that are encrypted using an approved security function. CSPs encrypted or obfuscated using non-approved security functions are considered unprotected plaintext within the scope of this International Standard.

ISO/IEC 24759:2017 AS02.21: (Specification — Levels 1, 2, 3, and 4) Non-approved cryptographic algorithms, security functions, and processes or other services not specified in {ISO/IEC 19790:2012} 7.4.3 shall not be utilized by the operator in an approved mode of operation unless the non-approved cryptographic algorithm or security function is part of an approved process and is not security relevant to the approved processes operation (e.g. a non-approved cryptographic algorithm or non-approved generated key may be used to obfuscate data or CSPs but the result is considered unprotected plaintext and provides no security relevant functionality until protected with an approved cryptographic algorithm).

Question/Problem

The term *non-approved security function* is not defined in the **ISO/IEC 19790:2012** Terms and Definitions, but is cited in multiple places in the standard, DTR and IG. How is *non-approved security function* defined, and how is it interpreted in relation to the **ISO/IEC 19790:2012** Section 7.2.4 Modes of operations?

Resolution

Definition of *non-approved security function*

FIPS 140-3 is concerned specifically with approved and non-approved security functions: the term *non-approved security function* must be defined relative to functions that claim security, rather than all functionality outside the set of *approved security functions*. The term *security* is not defined in the Terms and Definitions, but, within the scope of FIPS 140-3, is determined based on the Section 6 Functional Security Objectives, and the specific Section 7 Security Requirements derived from those objectives.

A *non-approved security function* is any function within the scope of the module that relies on a non-approved cryptographic algorithm to support a claim of security.

Notes

Per the definition of a cryptographic algorithm (see Background), primitive computational and logical operations (e.g. addition, subtraction, multiplication, division, AND, NOT, OR, and XOR) are used in cryptographic algorithms but are not themselves cryptographic algorithms.

A non-approved cryptographic algorithm or proprietary cryptographic algorithm is not a security function if processed data can be treated as plaintext without violating the Objectives stated in **ISO/IEC 19790:2012** Section 6, the applicable requirements in **ISO/IEC 19790:2012** Section 7, or the security rules specified in the module's Security Policy.

Relationship of non-approved cryptographic algorithms and the modes of operation

Non-approved security functions **shall not** be used in the approved mode of operation; however, non-approved cryptographic algorithms may be used in the approved mode of operation if the non-approved algorithms are not a security function. If a non-approved cryptographic algorithm is used by the module in the approved mode but is not a security function, the algorithm **shall** be included in the list of non-approved but allowed algorithms in the Security Policy with the caveat “(no security claimed)” (see **SP 800-140B** Section B.2.2). However, the module’s certificate **shall** not include these algorithms as they are not used to meet any requirement of FIPS 140-3.

A non-approved cryptographic algorithm **shall not** share the same key or CSP that is used by an approved or allowed algorithm for any cryptographic operation in either the approved, or non-approved mode, as this counters Section 6 Security Objectives by potentially releasing sensitive data and/or CSP(s). A non-approved cryptographic algorithm may still access or modify a CSP in the approved mode (under strict conditions laid out in this IG), as long as the CSP is not used as part of the non-approved cryptographic operation, such encryption/decryption, SSP establishment (inclusive of key generation), message authentication, message digest generation or digital signature generation/verification. The only exception to the rule explained in the first sentence of this paragraph, is the use of a non-approved cryptographic algorithm that utilizes an approved DRBG for any purpose such as SSP establishment, stand-alone random number generation, hashing, data obfuscation, etc. Despite access and modification of the state of the DRBG CSP(s) by a non-approved algorithm, this is allowed in both the approved and non-approved modes of operation. See the examples below for more information.

Possible example scenarios of non-approved cryptographic algorithms in various modes of operation

Example scenarios of non-approved cryptographic algorithms allowed in the approved mode

1. Use of a non-approved cryptographic algorithm to “obfuscate” a CSP

For purposes of storage or certificate formatting (e.g. PFX), a module might:

- XOR a CSP with a secret value
- Encrypt or decrypt a CSP using a proprietary or non-approved cryptographic algorithm.
- Store authentication data using MD5 or using HMAC-SHA-1 with a weak HMAC key
- Format certificate data using a non-approved PKCS #12

As noted above, “CSPs encrypted or obfuscated using a non-approved [algorithms] or proprietary algorithm or method are considered unprotected plaintext.”

All Section 7 requirements must be satisfied when considering the CSP in plaintext form:

- The report description of CSPs must correctly describe the form of the CSP.
- The module must support zeroization of any CSPs stored internally in the forms described above.
- If the obfuscated CSP is imported or exported, the module must meet the requirements for plaintext CSP import or export.

This conclusion is consistent with [IG 9.6.A Acceptable Algorithms for Protecting Stored Keys and CSPs](#).

2. Use of an approved, non-approved or proprietary algorithm for a purpose that is not security relevant or is redundant to an approved cryptographic algorithm

- a. Use of MD5 in the TLS 1.0 / 1.1 KDF

SP 800-135 Rev1 Section 4.2.1 describes the use of MD5 in conjunction with SHA-1 in the key derivation function, concluding that the TLS 1.0/1.1 KDF may be used within the context of the TLS protocol (with provisions for validation of the companion approved functions, SHA-1 and HMAC).

This use of MD5 does not conflict with the security of the approved security functions.

- b. Storage device use of a PRF (e.g. XTS AES) for memory wear leveling (a technique for prolonging the service life of some kinds of erasable computer storage media). For best results, a method with good statistical properties (i.e. a PRF) may be used for wear leveling, redundant to any other encryption or decryption performed by the module. This use of an algorithm is not for a security purpose; it is to prolong memory life.

- c. A secure channel operated over an insecure communications channel

Consider a module whose purpose is to provide end-to-end secure communications over an insecure communications channel. That channel may be plaintext or some method which provides insufficient security, assumed to provide no greater security than plaintext.

Specifically, assume the module communicates over a normal, unprotected Ethernet, provides approved end to end encryption, decryption and message authentication, as well as initial authentication of the peer node, and meets all FIPS 140-3 Section 7 requirements. This module can be validated.

Consider the same scenario but with wireless communications over WEP, WPA, WPA2 or similar, where the purpose of the module is a *remedy* for insecure communications media. The module must communicate with a WAP using the communications protocols the WAP provides. If the channel is treated as plaintext, and the module provides secure channel services that meet all FIPS 140-3 Section 7 requirements, to deny validation to such a module because the communications media uses non-approved functions defeats the purpose of the module, and is contrary to the intent of the CMVP as a program.

- d. Non-approved cryptographic algorithm that uses an approved DRBG for cryptographic purposes

The module uses a non-approved cryptographic algorithm to “obfuscate” a CSP for RAM storage. The key used for “obfuscation” is derived via an approved DRBG. By doing this, the DRBG changes its state, and therefore the DRBG CSPs are modified. Despite the modification and use of the DRBG CSPs within a cryptographic operation, this is allowed because the DRBG is the exception to the rule laid out in this IG.

- 3. Use of a non-approved cryptographic algorithm as part of an approved algorithm that claims security

- a. Use of GHASH within AES GCM

Although GHASH, alone, is a non-approved hashing function, it is used within an approved AES GCM algorithm, and is therefore permitted, even if the vendor claims security on this algorithm. However, if the vendor claims security on this function, then it **shall not** be used in the approved mode for any independent operation outside of the approved algorithm.

Example scenarios of non-approved cryptographic algorithms not allowed in any mode

- 1. Non-approved cryptographic algorithm that share the same key or CSP as an approved algorithm

- a. A DES algorithm is encrypting data using a DES key K1. This key is a part of a Triple-DES key $K = (K1, K2, K3)$ which is a CSP, as it may be used by an approved Triple-DES algorithm. The value $E = \text{DES}_{K1}(\text{data})$ is sent outside the module’s boundary. An attacker can easily break the single-DES encryption and recover K1, which will lead to the disclosure of the Triple-DES key K.
- b. Suppose a module generates, in full compliance with **FIPS 186-4**, a key pair for an approved RSA signature algorithm. However, the module also has a non-approved RSA signature algorithm not claiming any security. This non-approved RSA signature algorithm could use the same RSA key to generate its “signatures”. These non-approved signatures may be broken by an attacker and the signing key may be recovered, allowing the attacker to use this key to sign what *they* want.

The reason the above two examples are prohibited is because they do not follow the above rule which states: “A non-approved cryptographic algorithm **shall not** share the same key or CSP that is used by an approved or allowed algorithm for any cryptographic operation in

either the approved, or non-approved mode”. Even if the vendor claims no security on these non-approved algorithms, they are still not allowed.

Additional Comments

The vendor must provide clear documentation and reasoning as to why the non-approved cryptographic algorithms can be used in an approved mode, i.e. not being used to meet the requirements of FIPS 140-3 sections 6 and 7. It is at the discretion of the CMVP to determine if such usage of an algorithm fits within the guidance laid out in this IG.

2.4.B Tracking the Component Validation List

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

In response to vendor and user requirements, the CAVP has identified several components of the approved algorithms that they can test. When these components are successfully tested the vendor is issued the CVL (Component Validation List) certificates.

The reasons for introducing and testing these algorithm components differ. It can be that the module performs key agreement compliant to **SP 800-56Arev3**, but the shared secret computation, key derivation and optional key confirmation procedures of the key agreement scheme are tested individually rather than as one complete test, as approved by [IG D.F.](#)

In another example, the module may perform a cryptographic signature generation computation without computing the hash of the message as this hash has already been precomputed by another entity. Component testing allows one to verify the correctness of the remaining portion of the signature-generating routine.

Question/Problem

How to find the available testable components of the approved algorithms? Which documents specify the functions that each of these components performs?

Resolution

The following components can be tested and documented as CVLs in the module’s validation certificate.

1. An RSA (PKCS1-v1.5 and PSS) or ECDSA signature generation per **FIPS 186-4** without the computation of a hash which is presumed to have already been computed.

For RSA, the test verifies the correctness of the RSA exponentiation when performed as part of the digital signature generation. The test uses the integers m , d and n , where n is an RSA modulus, d plays the role of the private RSA key and m stands for the quantity based on the message to be signed, M , the selected approved hash function and the chosen RSA signature scheme (PKCS1-v1.5 or PSS). The primitive computes $s = m^d \bmod n$ and is described in PKCS#1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002, Section 5.2.1, step 2a. If the s value is successfully verified, the test passes.

There is also a test for a signature generation component (no hash computation) using the Chinese Remainder Theorem (CRT). This method of signature generation is described in the same standard, Section 5.2.1, step 2b.

For the ECDSA signature generation component, the test is the same as when the full ECDSA signature generation algorithm is tested except that the supplied messages are viewed as being already hashed, therefore no further hashing is performed. A binary string representing the hash is supplied to the test. The length of the supplied string is not tested for being valid. For details, please see the following CAVP publication: <https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/dss2/ecdsa2vs.pdf>, Section 6.4.1.

2. An RSA decryption operation using an exponentiation for key encapsulation, as specified in the section 7.1.2.1 of **SP 800-56Br2** published in March 2019.

As of September 21, 2020, there is no test for the decryption operation using the CRT, as shown in Section 7.1.2.3.

3. The key derivation functions from the following protocols and standards documented in **SP 800-135 Rev 1**: IKEv1, IKEv2, TLS 1.0, 1.1 and 1.2, SSHv2, SRTP, SNMPv3, TPMv1.2, ANSI X9.63-2001 KDF and ANSI X9.42-2001 KDF.
4. The TLS 1.3 key derivation function documented in [Section 7.1](#) of RFC 8446. This is considered an approved CVL because the underlying functions performed within the TLS 1.3 KDF map to NIST approved standards, namely: **SP 800-133rev2** (Section 6.3 Option #3), **SP 800-56Crev2**, and **SP 800-108**.
5. The key confirmation functionality described in the standards for the key agreement and key transport. The key confirmation can be unilateral or bilateral. See Sections 5.9 and 6.3.3 of **SP 800-56A Rev3** and Sections 5.6, 8.2.3, 8.3.3 and 9.2.4 of **SP 800-56B Rev2**. Key confirmation may be tested as a stand-alone function or as part of an end-to-end testing of a SSP establishment scheme. In the former case, a tested key confirmation is documented as a CVL.

The Security Policy **shall** individually list the tested components shown in the module's CVL certificates that may be called during the operation of the module.

Additional Comments

1. The testing of compliance to **SP 800-56A Rev3** will consist of testing of each of the shared secret computation schemes defined in Section 6 of this standard and implemented by the module. While **SP 800-56A Rev3** further shows how to apply the key derivation functions defined in **SP 800-56C Rev1**, the computation of a shared secret is viewed as a core functionality defined in **SP 800-56A Rev3**. Therefore, testing of this computation is not viewed as "component testing". If an implementation successfully passes these tests, it will be awarded an algorithm certificate, KAS-SSC, rather than a CVL certificate. This IG does not cover the KAS-SSC testing.
 2. At this time, no algorithm components are selected for vendor affirmation. This might change, as the CMVP may start giving vendors an opportunity to affirm the correct implementation of a component of a cryptographic algorithm where the entire approved algorithm has not been implemented in the module.
 3. The details of the CAVP component testing are provided at <https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program/Component-Testing>.
 4. Refer to [IG 10.3.A](#) for the applicability of self-tests to the tested components that have been issued the CVL certificates.
 5. **SP 800-135rev1** and the TLS 1.3 KDF are considered approved CVLs only when performed in the context of their respective protocols.
 6. **SP 800-140D** will be updated to include RFC 8446 Section 7.1, **SP 800-56Crev2** and **SP 800-133rev2** on its next release.
-

Section 3 – Cryptographic module interfaces

3.4.A Trusted Channel

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS03.16, AS03.17, AS03.18 AS03.19, AS03.20, AS03.21, AS03.22, AS06.27, AS09.20. AS09.21, ASA.01, ASB.01</i>
Relevant Test Requirements:	<i>TE's associated with AS's above</i>
Relevant Vendor Requirements:	<i>VE's associated with AS's above</i>

Background

ISO/IEC 19790:2012 Terms and Definitions:

Trusted channel: trusted and safe communication link established between the cryptographic module and a sender or receiver to securely communicate unprotected plaintext CSPs, key components and authentication data. NOTE: A trusted channel protects against eavesdropping, as well as physical or logical tampering by unwanted operators/entities, processes or other devices, between the module's defined input or output ports and along the communication link with the intended endpoint.

Key component: parameter used in conjunction with other key components in an approved security function to form a plaintext CSP or perform a cryptographic function.

ISO/IEC 19790:2012:

Table 1: Manually established SSPs may be entered or output in either encrypted form, via a trusted channel or using split knowledge procedures.

Section 7.3.4: A trusted channel is a link established between the cryptographic module and a sender or receiver to securely communicate unprotected plaintext CSPs, key components and authentication data. A trusted channel protects against eavesdropping, as well as physical or logical tampering by unwanted operators/entities, processes or other devices, between the module's defined input or output ports and along the communication link with the intended sender or receiver endpoint.

7.3.4 Trusted channel

- Security Levels 1 and 2
 - there are no requirements for a trusted channel.
- For Security Levels 3 and 4
 - for the transmission of unprotected plaintext CSPs, key components and authentication data between the cryptographic module and the sender's or receiver's endpoint the cryptographic module **shall [03.16]** implement a trusted channel;
 - the trusted channel **shall [03.17]** prevent unauthorised modification, substitution, and disclosure along the communication link;
 - the physical ports used for the trusted channel **shall [03.18]** be physically separated from all other ports or the logical interfaces used for the trusted channel **shall [03.19]** be logically separated from all other interfaces;

- identity-based authentication **shall [03.20]** be employed for all services utilising the trusted channel; and
- a status indicator **shall [03.21]** be provided when the trusted channel is in use.
- Security Level 4:
 - Multi-factor identity-based authentication **shall [03.22]** be employed for all services utilising the trusted channel.

7.6.3 Operating system requirements for modifiable operational environments

- Security Level 2 only
 - the audit mechanism of the operating system **shall [06.27]** be capable of auditing the following operating system related events:
 - attempts to use the trusted channel function and whether the request was granted, when trusted channel is supported at this security level; and
 - identification of the initiator and target of a trusted channel, when trusted channel is supported at this security level.

7.9.5 Sensitive security parameter entry and output

- For Security Level 3+
 - CSPs, key components and authentication data **shall [09.20]** be entered into or output from the module either encrypted or by a trusted channel.
 - CSPs which are plaintext secret and private cryptographic keys **shall [09.21]** be entered into or output from the module using split knowledge procedures using a trusted channel.

A.2.3 Cryptographic module interfaces (minimum documentation which **shall [A.01]** be required)

- For Security Levels 3+
 - Specification of the trusted channel interface.

B.2.3 Cryptographic module interfaces (requirements that **shall [B.01]** be provided in the non-proprietary security policy)

- Specify (each) trusted channel.

Question/problem

1. Is a trusted channel permitted at Security Levels 1 and 2?
2. What are the Security Policy requirements when using a Trusted Channel?

Resolution

The use of Trusted Channel is only applicable to **manual** SSP entry/output methods, per ISO 7.9.4 and 7.9.5 (see [IG 9.5.A](#) for examples of manual and automated methods). The security mechanism is physical protection, the operator having control over the physical path and is able to prevent any unauthorized tampering. The implementation of a trusted channel must consider the physical connection up to each port interface and the interface to the module. A dedicated port is not typically available for a processor-based system as the ports are usually multiplexed. The interface to the module, except in rare circumstances, is typically logical in nature.

The vendor **shall** describe how the physical connection either protects communications or is protected by trusted surrounding elements. The vendor **shall** also demonstrate that the logical connection can adequately gate the communications to deliver it to the module in a constrained manner. Logical communications through the physical connections cannot depend on encryption to create the trusted channel.

1. The use of a trusted channel is permitted at Security Levels 1 and 2 but **shall** meet the Security Level 3 requirements for paragraphs 7.3.4, A.2.3 and B.2.3. A module using a trusted channel in a modifiable operational environment **shall** also meet 7.6.3.
2. The Security Policy **shall** specify the following:
 - the physical characteristics of the Trusted Channel, with an explanation of how the Trusted Channel will protect the plaintext CSPs,
 - the controls that are used to maintain the Trusted Channel, including the list of any physical tools (wires, cables, etc.) needed to establish the Trusted Channel,
 - operator instructions for setup and operation of the Trusted Channel,
 - the specific characteristics and specification of the source or target of the Trusted Channel relative to the cryptographic module.

Additional comments

1. [IG 9.5.A](#) provides various scenarios that apply to both physical and cryptographic protection of CSPs when they are either entered as input or output out of the module's boundary, with several examples of physical devices that can be used in CSP entry or output.
2. It is possible for a module to get validated at different security levels in Sections 7.3 and 7.9 of **ISO/IEC 19790:2012**, as these sections are addressing the different sets of requirements. For example, a module can meet the Security Level 3 requirements of Section 7.3 by inputting the plaintext cryptographic keys using the Trusted Channel provided by a directly attached cable. However, this module will only be validated at Security Levels 1 or 2 in Section 7.9, as the imported keys are neither encrypted nor entered in plaintext using the split knowledge procedures using a trusted channel. This example is consistent with the fact that the requirements of Section 7.9 are stricter than those of Section 7.3.

Section 4 – Roles, services, and authentication

4.1.A Authorised Roles

Applicable Levels:	<i>2, 3, and 4</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	
Relevant Test Requirements:	
Relevant Vendor Requirements:	

Background

ISO/IEC 19790:2012 Section 7.4.1:

An operator is not required to assume an authorised role to perform services where CSPs and PSPs are not modified, disclosed, or substituted (e.g. show status, self-tests, or other services that do not affect the security of the module).

Authentication mechanisms may be required within a cryptographic module to authenticate an operator accessing the module, and to verify that the operator is authorised to assume the requested role and perform the services within the role.

Question/Problem

What are the services that do not require an operator, in the approved mode, to assume an authorised role and, therefore, not be authenticated, as required if Security Level 2, 3, or 4 is claimed for Section 7.4?

Resolution

If a Security Level 2 or above is claimed for Section 7.4, an operator in the approved mode **shall** be authenticated when assuming a role for all services utilizing approved security functions, with the following exceptions:

- The hash algorithms which are specified in [FIPS 180-4](#) and [FIPS 202](#);
- The deterministic random number generators which are specified in [SP 800-90A rev1](#). If the DRBG service is provided to an authenticated operator, the entropy source seeding the DRBG **shall** be completely contained within the boundary of the cryptographic module;
- Digital signature verification, as specified in "Digital Signature Standard", [FIPS 186-2](#) and [FIPS 186-4](#).
- Authentication procedures used for authenticating the operator and/or initialization procedures to setup the operator's authentication credentials; and
- Show status, show version, self-tests, or other services that do not affect the security of the module (where CSPs and PSPs are not modified or substituted, and CSPs are not disclosed)

Additional Comments

1. The reason for the stated exceptions is that the referenced algorithms do not create, disclose or modify the module's CSPs.
2. **ISO/IEC 19790:2012** Section 7.4 talks about "authorised" roles. For the purposes of this IG, an authorised role is any defined role. Some of these defined roles may require an operator to get authenticated before the operator is authorised to assume the role.
3. Performing any service requires an assumption of a role. This IG clarifies under what conditions some of the roles may remain unauthenticated. When the **ISO/IEC 19790:2012** standard states (see the **Background** section above) that an operator is not required to assume an authorised role to perform certain services, this means that while the module may be validated at Security Level 2 or above in Section 7.4, a defined role may not require an authentication of an operator for the role to perform these services.
4. It is stated in (e) in the Resolution section that an unauthenticated service may not, at Security Levels 2 and above, cause a modification of the module's SSPs. There exists an exception to this rule. An approved DRBG may be called from an unauthenticated role, or even from a role that includes the non-approved services. Each execution of a DRBG may result in a modification of the DRBG's secret state parameters, which are the module's CSPs (see [IG D.L](#)). This indirect modification of the CSPs is permissible because it does not result in the weakening of the CSPs or in a loss of their secrecy.
5. The zeroization of all of the module's unprotected SSPs performed as required in Section 7.9.7 of **ISO/IEC 19790:2012** is not viewed as a "modification" of these parameters. Therefore, the corresponding zeroisation service may be called from an unauthenticated role.

4.4.A Multi-Operator Authentication

Applicable Levels:	<i>Security Levels 2, 3 and 4</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS04.57, AS04.58, AS04.59</i>
Relevant Test Requirements:	<i>TE's associated with AS's above</i>
Relevant Vendor Requirements:	<i>VE's associated with AS's above</i>

Background

ISO/IEC 19790:2012 Section 7.4.4 Authentication

Authentication mechanisms may be required within a cryptographic module to authenticate an operator accessing the module and to verify that the operator is authorised to assume the requested role and perform services within that role. The following types of mechanisms are used to control access to the cryptographic module:

- a) *Role-Based Authentication*: If role-based authentication mechanisms are supported by a cryptographic module, the module **shall [04.36]** require that one or more roles either be implicitly or explicitly selected by the operator and **shall [04.37]** authenticate the assumption of the selected role (or set of roles). **The cryptographic module is not required to authenticate the individual identity of the operator.** The selection of roles and the authentication of the assumption of selected roles may be combined. If a cryptographic module permits an operator to change roles, then the module **shall [04.38]** authenticate the assumption of any role that was not previously authenticated for that operator.
- b) *Identity-Based Authentication*: If identity-based authentication mechanisms are supported by a cryptographic module, the module **shall [04.39]** require that the operator be individually and uniquely identified, **shall [04.40]** require that one or more roles either be implicitly or explicitly selected by the

operator, and **shall [04.41]** authenticate the identity of the operator and the authorisation of the operator to assume the selected role or set of roles. The authentication of the identity of the operator, selection of roles, and the authorisation of the assumption of the selected roles may be combined. If a cryptographic module permits an operator to change roles, then the module **shall [04.42]** verify the authorisation of the identified operator to assume any role that was not previously authorised.

AS04.57: (Operator authentication — Level 2) A cryptographic module shall at a minimum employ *role-based authentication* to control access to the module.

AS04.58: (Operator authentication — Levels 3 and 4) A cryptographic module shall employ *identity-based authentication mechanisms* to control access to the module.

AS04.59: (Operator authentication — Level 4) A cryptographic module shall employ *multi-factor identity-based authentication mechanisms* to control access to the module.

Question/Problem

A module may implement separately defined operator roles which have different authentication claims. For example, the Crypto Officer (CO) role implements *identity-based authentication* while the User role implements *role-based authentication* (Case 1). In another example, the CO role implements *role-based authentication* while the User role does not implement any *authentication* (Case 2). There is also a possibility of the CO and User roles each supporting role-based as well as the identity-based authentication (Case 3): some of the operators who are assuming a given role are authenticated using the role-based credentials, while others, who will also assume this role, pass an identity-based authentication. In addition, the Crypto Officer (CO) role may implement *identity-based authentication* while the User role implements *multi-factor identity-based authentication* (Case 4). Are these implementations compliant with the requirements of Section 7.4.4 of **ISO/IEC 19790:2012**, and, if so, at what security level?

For the above scenarios, it is assumed that approved security services are included in each assumed role. Should there be an exception to the operator authentication requirement when the approved security functions do not affect the security of the module?

Resolution:

Following are the resolutions for the three scenarios from the Question/Problem section above.

1. The first case (Case 1) is compliant to **ISO/IEC 19790:2012** Section 7.4.4 because for the purposes of the FIPS 140-3 validation, *identity-based authentication* is considered to be meeting the *role-based authentication* requirement. Both the CO and the User operators get authenticated to access the approved security services. The section Security Level is 2 because it is the lower of the two authentication methods described.

The Security Policy **shall** identify all roles, and for each role, the authentication method (i.e. either *role-based* or *identity-based*).

2. In the second case (Case 2) the module is compliant to **ISO/IEC 19790:2012** Section 7.4.4 Security Level 2 only if the unauthenticated User role does not call any services that affect the module's security. See [IG 4.1.A](#) for the definition of such services. Otherwise, **ISO/IEC 19790:2012** Section 7.4 is annotated at Security Level 1 and only the Security Level 1 assertions are addressed.
3. The Case 3 scenario is also compliant with FIPS 140-3. The vendor can claim compliance with Section 7.4 only at Security Level 2. The test report addresses each role at Security Level 2. The Security Policy **shall** explain how the authentication may be performed for each role.
4. The Case 4 scenario is compliant to **ISO/IEC 19790:2012** Section 7.4.4 because for the purposes of the FIPS 140-3 validation, *multi-factor identity-based authentication* is considered to be meeting the *identity-based authentication* requirement. Both the CO and the User operators get authenticated to access the

approved security services, but the User uses multi-factor authentication methods. The section Security Level is 3 because it is the lower of the two authentication methods described.

The Security Policy **shall** identify all roles, and for each role, the authentication method (i.e. either *multi-factor identity-based* or *identity-based*).

Additional Comments

1. [IG 4.1.A](#) addresses authenticated roles for approved security services and non-authenticated services.
2. In Case 3, the module can only be validated at Security Level 2 in Section 7.4 because the role-based authentication is also available to the module. Similarly, in Case 4, the module can only be validated at Security Level 3 in Section 7.4 because the identity-based authentication is also available to the module.
3. Other mixed cases are also possible. There is sufficient information in this Implementation Guidance to determine how to treat each of these cases and what will be the overall security level of the module's validation in Section 7.4. For example, the User role can have both a role-based and an identity-based authentication, while the Crypto Officer role always requires an identity-based authentication. As shown above, such a module is validated at Security Level 2 in Section 7.4, unless the User role only calls the services that are exceptions identified in [IG 4.1.A](#) as not affecting the module's security. If the latter case, the module's Section 7.4 may be validated at Security Level 3.
4. When the module supports both the role-based and the identity-based authentication, either within the same role (as in Case 3 above) or by the different roles (as in Case 1), the testing laboratory, when writing the Test Report, **shall** select the identity-based authentication option on the website form. This will require the testing laboratory to address in the test report both the Security Level 2 (role-based) and the Security Level 3 (identity-based) assertions. Similarly, multi-factor identity authentication used at any level will require the testing laboratory to address in the test report Security Level 4 (multi-factor identity-based) assertions.

Section 5 – Software/Firmware security

5.A Non-Reconfigurable Memory Integrity Test

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS05.05 to AS05.23, AS10.17, AS10.18,</i>
Relevant Test Requirements:	<i>TE's associated with AS's above</i>
Relevant Vendor Requirements:	<i>VE's associated with AS's above</i>

Background

ISO/IEC 19790 Section 7.10.2.2:

All software and firmware components within the cryptographic boundary **shall [10.17]** be verified using an approved integrity technique satisfying the requirements defined in [Section] 7.5. If the verification fails, the preoperational software/firmware integrity test **shall [10.18]** fail. The pre-operational software/firmware integrity test is not required for any software or firmware excluded from the security requirements of this International Standard or for any executable code stored in non-reconfigurable memory.

Question/Problem

What is the definition of “non-reconfigurable memory”?

Resolution

Non-reconfigurable memory **shall** be defined as a memory technology that stores data using a mechanical means (e.g. masked ROM, CD-ROM) that will not change or degrade once manufactured for a minimum of 20 years.

The tester **shall** verify (in TE05.05.01 or TE05.06.01) that vendor provided documentation describes how there will be no change or degradation of data for a minimum of 20 years.

The software or firmware integrity test is not required for executable code stored in non-reconfigurable memory. This code is considered hardware.

Additional Comments

The reason for the above definition on what constitutes non-reconfigurable memory is that most common read-only memory technologies (e.g. OTP, PROM, WORM, CD-R) store data using a chemical change or electrical charge that will likely degrade at a higher rate than data stored using mechanical means. This IG **shall not** apply to these cases and the memory would still be subject to the integrity test.

Section 6 – Operational environment

Section 7 – Physical security

7.3.A Testing Tamper Evident Seals

Applicable Levels:	<i>Levels 2,3 and 4</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS07.48</i>
Relevant Test Requirements:	<i>TE07.48.02</i>
Relevant Vendor Requirements:	<i>VE07.48.02</i>

Background

ISO/IEC 24759:2017:

AS07.48: (Multi-chip embedded cryptographic modules – Level 2,3, and 4)

{If AS07.45 is not satisfied and the enclosure includes any doors or removable covers without matching AS07.47, then they (i.e. the doors or removable covers)} **shall** be protected with tamper evident seals (e.g. evidence tape or holographic seals) {and the group (AS07.47 and AS07.49) shall be satisfied}.

TE07.48.02: The tester shall verify that the cover or door cannot be opened without breaking or removing the seal and that the seal cannot be removed and later replaced.

Question/Problem

What level of testing and scope of testing should be applied when testing tamper evident seals?

Resolution

If a module uses tamper evident labels, it **shall** not be possible to remove or reapply any of the labels without tamper evidence. For example, if the label can be removed without tamper evidence, and the same label can be re-applied without tamper evidence, the assertion fails. Note at level 3 and 4 **AS07.27** requires tamper seals be independently identifiable to make it harder to replace without tamper evidence; testing that is outside the scope of this IG.

Conversely, if any attempt to remove the label leaves evidence, or removal and re-application leaves evidence, or the label is destroyed during removal, the assertion passes. If the label placement documented in the Security Policy does not match the placement of the tamper seals on the module under test, the removal or destruction of a label would be evident and considered evidence of tampering.

This means that the CST laboratory **shall** use creative ways (e.g. chemically, mechanically, thermally) to remove a label without evidence and without destroying the original label and be able to re-apply the removed label in a manner that does not leave evidence.

Additional Comments

It is out-of-scope for an attacker to introduce new materials to cover up evidence of the attack.

7.3.B Hard Coating Test Methods (Level 3 and 4)

Applicable Levels:	<i>Level 3 and 4</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS07.26, AS07.35, AS07.37, AS07.41, AS07.42</i>
Relevant Test Requirements:	ISO/IEC 24759:2017 <i>TE07.37.01, TE07.37.02, TE07.37.03, TE07.37.04, TE07.41.01, TE07.41.02, TE07.42.01, TE07.42.02</i> SP 800-140 <i>TE07.26.01, TE07.26.02</i>
Relevant Vendor Requirements:	SP 800-140 <i>VE07.26.01, VE07.26.02</i>

Background

ISO/IEC 19790:2012 Terms and Definitions:

Hard / hardness: the relative resistance of a metal or other material to denting, scratching, or bending; physically toughened; rugged, and durable.

Note: The relative resistances of the material to be penetrated by another object.

ISO/IEC 24759:2017:

AS07.26: (Physical security — Levels 3 and 4) Strong or hard conformal or non-conformal enclosures, coatings, or potting materials shall maintain strength and hardness characteristics over the module's intended temperature range of operation, storage, and distribution.

AS07.37: (Single-chip cryptographic modules — Levels 3 and 4) {Either} the module shall be covered with a hard opaque tamper-evident coating (e.g. a hard opaque epoxy covering the passivation) {or AS07.38 shall be satisfied}.

SP 800-140:

TE07.26.01: The tester shall verify from the vendor documentation and inspection testing of the module that the strength or hardness of the, hard conformal or non-conformal enclosure, coatings or potting materials is the one designed implemented as specified. The tester shall verify the module hardness at the following temperatures:

- the lowest temperature of the module's intended temperature range of operation, storage and distribution;
- the highest temperature of the module's intended temperature range of operation, storage and distribution.

TE07.26.02: The tester shall verify that the vendor provided Security Policy specifies the high/low temperature range.

Question/Problem

What kind of testing is expected to be performed at Security Level 3 to verify that the hard coating or potting material that encapsulates the circuitry is *hard*?

Resolution

Test methods **shall** address a *moderately aggressive attack* at Security Level 3.

The test methods **shall** at a minimum address the hardness characteristics of the epoxy or potting material as follows:

1. Attempts to penetrate the material by an instrument (e.g. awl, pointed handheld tool, etc.) using a *moderately aggressive* amount of force to the depth of the underlying circuitry. The use of a drilling or grinding motion is out-of-scope.
2. The use of an instrument with a *moderately aggressive* amount of force to pry or break the material away from the underlying circuitry (e.g. insert a pry instrument at the boundary of the epoxy or potting material and another material/component (e.g. PCB board)).
3. The use of a *moderately aggressive* amount of flexing or bending force to crack or break the material away from or expose the underlying circuitry.

During testing the module should be consistently assessed to determine if serious damage has occurred (i.e. the module will either cease to function or the module is unable to function).

The epoxy or potting material **shall** be tested to determine if voids or pockets may exist that could create an exposure or weakness.

Module hardness testing **shall** be performed using calibrated equipment at the module's intended temperature range of operation, storage and distribution.

Additional Comments

While the above test methods may be applicable at *Physical Security* Level 3 for a module which is protected by a strong enclosure or includes doors or removable covers, this IG does not specifically address those test methods.

Section 8 – Non-invasive security

Section 9 – Sensitive security parameter management

9.3.A Entropy Caveats

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.08</i>
Relevant Test Requirements:	<i>TE09.08.01-2</i>
Relevant Vendor Requirements:	<i>VE09.08.01-2</i>

Background

Section 7.9.3 of **ISO/IEC 19790:2012** states that “Compromising the security of the SSP generation method which uses the output of an approved RBG (e.g., guessing the seed value to initialise the deterministic RBG) **shall [09.08]** require at least as many operations as determining the value of the generated SSP.” TE09.08.02 further states that “The tester shall verify the accuracy of any rationale provided by the vendor. The burden of proof is on the vendor; if there is any uncertainty or ambiguity, the tester shall require the vendor to produce additional information as needed.”

There are some module designs where it may be impossible to know how much entropy has been supplied for key generation. For example, a module designed as a software library with an API allowing the caller to supply random buffer to use as a seed for random number generation, the module would be passively accepting the entropy “infusions” from third-party applications. From such module’s perspective, it is only possible to talk about the number of bytes/bits size of the received random field, not of the amount entropy in it. Does it mean that the requirement in **AS09.08** cannot be tested and therefore the module cannot be validated?

In this example, the module is not necessarily non-compliant with **AS09.08**; it is just impossible to determine (within the scope of the CST lab testing) that the module would be compliant in all possible deployments. This IG weighs this and similar issues and shows how to identify the cases when compliance with that entropy requirements of FIPS 140-3 cannot be directly verified by the testing labs and how to inform the user of potential weakness or lack of assurance for the true strengths of the SSPs generated by such modules.

Question/Problem

When is it necessary for the module to provide the evidence of the amount of generated entropy?

How to handle the case when the amount of generated entropy is sufficient to meet the minimum SSP strength requirement (112 bit) but not necessarily sufficient to account for a comparable strength of the generated SSPs?

What information **shall** the testing laboratory provide in the test report submitted to the CMVP?

What information **shall** be included in the module’s certificate and the Security Policy (SP) to indicate the various forms of compliance with the **AS09.08** requirement?

Resolution

We identify the main “logical” cases and for each case indicate whether the module can be validated and what certificate caveat, if any, **shall** be used.

1. The module is either generating the entropy itself or it is making a call to request the entropy from a well-defined source.

Examples include:

- (a) A hardware module with an entropy generating ENT inside the module's cryptographic boundary.

What is required: (i) the testing lab **shall** corroborate the entropy strength estimate as provided by the vendor, (ii) the SP **shall** state the minimum number of bits of entropy generated by the module for use in SSP generation.

If the amount of entropy used to generate the module's SSPs employed in an approved mode is less than 112 bits, then this module **cannot** be validated.

If the amount of entropy used to generate the module's SSPs is at least 112 bits while the module generates SSPs with a comparable cryptographic strength greater than the amount of the available entropy, the following caveat **shall** be included in the module's certificate: *The module generates SSPs whose strengths are modified by available entropy.* The comparable cryptographic strength of an SSP is addressed under the Additional Comments below.

- (b) A software module that contains an approved DRBG, that is seeded exclusively from one or more known entropy sources, located within the physical perimeter of the operational environment. For instance, a software library on a Linux platform making a call to a **SP 800-90B** entropy source within the module's physical perimeter for seeding its DRBG.

What is required: (i) the testing lab **shall** corroborate the entropy strength estimate of the sources as provided by the vendor, (ii) the SP **shall** state the minimum number of bits of entropy requested per each GET function call.

If the amount of entropy used to generate the module's SSPs employed in an approved mode is less than 112 bits, then this module cannot be validated.

If the amount of entropy used to generate the module's SSPs is at least 112 bits while the module generates SSPs with a comparable cryptographic strength greater than the amount of available entropy, the following caveat **shall** be included in the module's certificate: *The module generates SSPs whose strengths are modified by available entropy.*

- (c) A software module that contains an approved DRBG that issues a GET command to obtain the entropy from a source located outside the module's physical perimeter.

What is required: (i) the testing lab **shall** corroborate – to the extent it is possible, given that the entropy source is not subject to this module's testing and validation – the entropy strength estimate as provided by vendor, (ii) the SP **shall** state the minimum number of bits of entropy requested per each GET function call, (iii) the following caveat **shall** be added to the module's certificate: *No assurance of the minimum strength of generated SSPs.*

If the claimed amount of obtained entropy used to generate the module's SSPs employed in an approved mode is known to be less than 112 bits, then this module cannot be validated.

2. The module is passively receiving the entropy while exercising no control over the amount or the quality of the obtained entropy.

Examples include:

- (a) A hardware module with an approved DRBG inside the module's cryptographic boundary. The approved DRBG is either seeded via a seed loader from outside the module's cryptographic boundary or the seed is pre-loaded at factory.

What is required: (i) the SP **shall** state the minimum number of bits of entropy believed to have been loaded and justify the stated amount (from the length of the entropy field and from any other factors known to the vendor), (ii) the following caveat **shall** be added to the module's certificate: *No assurance of the minimum strength of generated SSPs.*

If the amount of claimed entropy used to generate the module's SSPs employed in an approved mode is known to be less than 112 bits, then this module cannot be validated.

- (b) A software module that contains an approved DRBG that receives a LOAD command (or its logical equivalent) with entropy obtained from either inside the physical perimeter of the operational environment of the module or, via an I/O port, from an external source that is outside the module's physical perimeter.

What is required: (i) the SP **shall** state the minimum number of bits of entropy believed to have been loaded and justify the stated amount (from the length of the entropy field and from any other factors known to the vendor), (ii) the following caveat **shall** be added to the module's certificate: *No assurance of the minimum strength of generated SSPs.*

If the amount of entropy used to generate the module's SSPs employed in an approved mode is *known to be* less than 112 bits, then this module cannot be validated.

- 3. The module uses a *hybrid* approach to obtaining entropy for SSP generation. Some entropy is passively received while the module is exercising no control over the amount or the quality of the obtained entropy. Another portion of the entropy is obtained when the module is either generating the entropy by itself or is making a GET call to request the entropy from a well-defined source inside the module's physical perimeter. For instance, a software library on a Linux platform may be making a call to /dev/random for seeding its DRBG while it is also providing an API allowing the calling application to supply an additional random buffer to use in seeding its DRBG.

What is required: The testing lab **shall** examine the design of seeding the DRBG from multiple sources and corroborate an entropy strength estimate as provided by vendor; the lab will need to understand the work of the ENT within the operational environment and be able to verify vendor's claim about the amount of entropy loaded into the software cryptographic module.

If the review of the design of seeding the DRBG reveals that the entropy data obtained passively can **only** add to the entropy obtained actively and the module will block the seeding until a minimal threshold amount of actively obtained entropy is reached, then

The SP **shall** state the minimum number of bits of entropy that can be guaranteed to be actively obtained and, in addition, it **shall** state the number of bits believed to have been loaded and justify the stated amounts (from the lengths of the entropy fields and from any other factors known to the vendor).

If between the active and passive entropy calls the module cannot possibly accumulate at least 112 bits of entropy when generating SSPs, then this module cannot be validated.

If the amount of entropy obtained actively may be less than 112 bits, then the following caveat **shall** be added to the module's certificate: *No assurance of the minimum strength of generated SSPs.*

If the review of the design of the DRBG seeding reveals that the entropy data obtained passively can preempt the seeding of the DRBG in a way that causes the module to unblock the seeding even when the minimal threshold amount of entropy obtained actively has not been reached at any time when the caller uses the API for supplying the passive data, then

The SP **shall** state the minimum number of bits of entropy believed to have been loaded and justify the stated amount (from the length of the entropy field and from any other factors known to the vendor).

If the module cannot possibly accumulate at least 112 bits of entropy when generating SSPs, then this module cannot be validated.

The following caveat **shall** be added to the module's certificate: *When entropy is externally loaded, no assurance of the minimum strength of generated SSPs.*

Additional Comments

1. Unless the design of the module falls under the case for which a specific caveat is explicitly allowed under a scenario described in this IG, the vendor may not use the caveat. In particular, the vendor cannot use the “*No assurance of the minimum strength of generated SSPs*” caveat and get their module validated if the scenario that applies to this module requires an explicit estimation of the generated entropy.
2. If a software module’s design requires entropy estimation then the module’s SP **shall** contain a statement that if porting to an untested platform is allowed then when running a module on such an untested platform the “*No assurance of the minimum strength of generated SSPs*” caveat applies regardless of what caveat, if any, is applicable to the original validation.
3. This implementation guidance only covers the applicability of entropy estimation and the way to document the amount of the available entropy. The actual methodology for entropy estimation is addressed in IGs [D.J](#) and [D.K](#).
4. Per **SP 800-57-part 1 revision 5** Section 5.6.1, the “comparable” strengths of security depend on the algorithm and the key size used and are based on accepted estimates as of the publication of this Special Publication using currently known methods. See [IG D.B](#) to determine the “comparable” cryptographic strength of different SSPs based on their length and known vulnerabilities.
5. If the module generates random strings that are not SSPs and the security strength of a generated string is less than the bit length of the string due to limited entropy, then the strength caveats shown in this IG are applicable, but they **shall** reference random strings rather than SSPs. For example, in scenario 1(b) above, the caveat would say: *The module generates random strings whose strengths are modified by available entropy*.

If the module generates both keys and random strings that have security strengths smaller than the presumed strengths of the keys and strings, then the caveat **shall** address the potential loss of strength in both keys and the random strings: *The module generates SSPs and random strings whose strengths are modified by available entropy*.

The module’s SP **shall** state the guaranteed amount of entropy for both the SSPs and the random strings generated by the module using the available entropy source(s).

6. There exist situations where it could be reasonable to place two different entropy caveats in the module’s validation certificate. For example, a software module receives a LOAD command that carries an externally generated entropy (scenario 2(b) above). The module uses this entropy to generate the 256-bit AES keys, yet the length of the received entropy string is, say, 192 bits. As shown above, this module may be validated. Since the entropy is generated externally, the *No assurance of the minimum strength of generated SSPs* caveat is required. In addition, the user can be certain that the obtained entropy is insufficient to generate an AES key with the 256-bit strength. Should the module’s certificate also include another available caveat: *The module generates SSPs whose strengths are modified by available entropy*?

The approach taken in this IG is that when more than one caveat might be needed, the module’s certificate **shall** document only the strongest caveat. In the above example, it is *No assurance of the minimum strength of generated SSPs*. The scenarios of this IG are written following this single-caveat approach.

The module’s SP **shall** inform the reader about the length of a random string loaded into the module and explain, if applicable, the effect of the random string length on the strengths of the generated keys.

Test Requirements

The vendor and tester evidence **shall** be provided under TE09.08.01 and TE09.08.02.

9.5.A SSP Establishment and SSP Entry and Output

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>

Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS03.16 – AS03.22, AS09.10 – AS09.24</i>
Relevant Test Requirements:	<i>TE's associated with AS's above</i>
Relevant Vendor Requirements:	<i>VE's associated with AS's above</i>

Background

ISO/IEC 19790:2012 Glossary:

Automated: without manual intervention or input (e.g. electronic means such as through a computer network).

Cryptographic boundary: explicitly defined perimeter that establishes the boundary of all components (i.e. set of hardware, software or firmware components) of the cryptographic module.

Direct entry: entry of a SSP or key component into a cryptographic module, using a device such as a keyboard.

Electronic entry: entry of SSPs or key components into a cryptographic module using electronic methods.

Hardware module interface (HMI): total set of commands used to request the services of the hardware module, including parameters that enter or leave the module's cryptographic boundary as part of the requested service.

Hybrid firmware module interface (HFMI): total set of commands used to request the services of the hybrid firmware module, including parameters that enter or leave the module's cryptographic boundary as part of the requested service.

Hybrid software module interface (HSMI): total set of commands used to request the services of the hybrid software module, including parameters that enter or leave the module's cryptographic boundary as part of the requested service.

Key component: parameter used in conjunction with other key components in an approved security function to form a plaintext CSP or perform a cryptographic function.

Key loader: self-contained device that is capable of storing at least one plaintext or encrypted SSP or key component that can be transferred, upon request, into a cryptographic module. NOTE The use of a key loader requires human manipulation.

Manual: requiring human operator manipulation.

Operational environment (OE): refers to the management of the software, firmware, and/or hardware required for the module to operate. The operational environment of a software, firmware, or hybrid module includes, at a minimum, the module components, the computing platform, and the operating system that controls or allows the execution of the software or firmware on the computing platform.

Plaintext key: unencrypted cryptographic key or a cryptographic key obfuscated by non-approved methods which is considered unprotected.

Software: executable code of a cryptographic module that is stored on erasable media which can be dynamically written and modified during execution while operating in a modifiable operational environment.

Software module: module that is composed solely of software.

Software/firmware module interface (SFMI): set of commands used to request the services of the software or firmware module, including parameters that enter or leave the module's cryptographic boundary as part of the requested service.

Split knowledge: process by which a cryptographic key is split into multiple key components, individually sharing no knowledge of the original key, that can be subsequently input into, or output from, a cryptographic module by separate entities and combined to recreate the original cryptographic key.

SSP establishment: process of making available a shared SSP to one or more entities. NOTE SSP establishment includes SSP agreement, SSP transport and SSP entry or output.

Trusted channel: trusted and safe communication link established between the cryptographic module and a sender or receiver to securely communicate unprotected plaintext CSPs, key components and authentication data. NOTE A trusted channel protects against eavesdropping, as well as physical or logical tampering by unwanted operators/entities, processes or other devices, between the module's defined input or output ports and along the communication link with the intended endpoint.

ISO/IEC 19790:2012 Section 7.9.4 Sensitive security parameter establishment:

SSP establishment may consist of

- automated SSP transport or SSP agreement methods or
- manual SSP entry or output via direct or electronic methods.

Automated SSP establishment **shall [09.10]** use an approved method listed in Annex D. Manual SSP establishment **shall [09.11]** meet the requirements of 7.9.5.

ISO/IEC 19790:2012 Section 7.9.5 Sensitive security parameter entry and output:

SSPs may be manually entered into or output from a module either *directly* (e.g. entered via a keyboard or number pad, or output via a visual display) or *electronically* (e.g. via a smart card/tokens, PC card, other electronic key loading device, or the module operating system). If SSPs are manually entered into or output from a module, the entry or output **shall [09.12]** be through the defined HMI, SFMI, HFMI or HSMI (7.3.2) interfaces.

To prevent the inadvertent output of sensitive information, two independent internal actions **shall [09.16]** be required in order to output any plaintext CSP.

For electronic entry or output via a wireless connection; CSPs, key components and authentication data **shall [09.18]** be encrypted.

SECURITY LEVELS 1 AND 2

Plaintext CSPs, key components and authentication data may be entered and output via physical port(s) and logical interface(s) shared with other physical ports and logical interfaces of the cryptographic module.

For software modules or the software components of a hybrid software module, CSPs, key components and authentication data may be entered into or output in either encrypted or plaintext form provided that the CSPs, key components and authentication data **shall [09.19]** be maintained within the operational environment and meet the requirements of 7.6.3.

SECURITY LEVEL 3

In addition to Security Levels 1 and 2, for Security Level 3, CSPs, key components and authentication data **shall [09.20]** be entered into or output from the module either encrypted or by a trusted channel.

CSPs which are plaintext secret and private cryptographic keys **shall [09.21]** be entered into or output from the module using split knowledge procedures using a trusted channel.

If the module employs split knowledge procedures, the module **shall [09.22]** employ separate identity-based operator authentication for entering or outputting each key component, and at least two key components **shall [09.23]** be required to reconstruct the original cryptographic key.

SECURITY LEVEL 4

In addition to Security Level 3, for Security Level 4 the module **shall [09.24]** employ multi-factor separate identity-based operator authentication for entering or outputting each key component.

Question/Problem

Given different configurations of cryptographic modules, how can a module's SSP establishment and SSP entry and output states be easily mapped to the **ISO/IEC 19790:2012** Section 7.3 *Cryptographic module interfaces*, Section 7.9.4 *Sensitive security parameter establishment* and Section 7.9.5 *Sensitive security parameter entry and output*? Are there any special considerations for *Sub-Chip Cryptographic Subsystems* ([IG 2.3.B](#))?

Resolution

Using the following guidelines, first determine how CSP keys¹ and non-keys (key components, authentication data, secret IVs, and other non-key CSPs) are established to or from a module using Table 1. Once the establishment method is determined, the CSP Entry Format table will indicate the requirements on how key and non-key CSPs **shall** be entered or output. The following is based on the requirements found in **ISO/IEC 19790:2012** in Sections 7.3 and 7.9.

CM: a FIPS 140-3 *validated* Cryptographic Module.

GPC: General Purpose Computer.

EXT: external/outside of the cryptographic boundary and tested operational environment (which may be the same for a hardware module).

INT: internal/inside of the cryptographic boundary.

TOEPP: external/outside of the cryptographic boundary but within the module's Tested Operational Environment's Physical Perimeter (TOEPP) (may apply to software, firmware, hybrid, and sub-chip modules).

App: general purpose software application operating outside the cryptographic boundary but inside of the operational environment (e.g. within the operating system).

CSP Establishment – Table 1	
MD: Manual Distribution	DE: Direct Entry (Input / Output)
AD: Automated Distribution	EE: Electronic Entry (Input / Output)
WD: Wireless Distribution	
MD / DE - CM from Keyboard	
CM Software ² from GPC keyboard or number pad	MD / DE
CM Hardware from non-networked GPC's keyboard or number pad	MD / DE
CM Hardware from directly attached keyboard or number pad	MD / DE
MD / DE – CM to Visual display	
CM Software ² to GPC visual display	MD / DE
CM Hardware to visual display or non-networked GPC's visual display	MD / DE
MD / EE – CM to/from EXT Hardware (key loader)	
CM Software ² to/from GPC key loader (e.g., smart card, CD, diskette, USB token, etc.)	MD / EE
CM Hardware to/from directly attached key loader (a non-networked GPC could be considered and used as a key loader)	MD / EE
CM Software ² running on a non-networked GPC to/from EXT CM Hardware (key loader)	MD / EE
CM Hardware to/from EXT CM Hardware via directly connected EXT Path (e.g. CM bound to another CM)	MD / EE
MD / EE – CM to/from TOEPP Path	
CM Software ² to/from CM Software ² via TOEPP Path (e.g. CM bound to another CM)	MD / EE
CM Software ² to/from App via TOEPP Path	MD / EE
AD / EE³ – CM to/from EXT Path via automated methods	
CM Software ² to/from GPC EXT using a network port	AD / EE
CM Software ² to/from CM Software ² via EXT Path	AD / EE

¹ An intermediate computational parameter, such as a shared secret in a key agreement scheme, is considered a key for the purposes of this Implementation Guidance if an actual key can be derived from this intermediate value without the knowledge of any other CSPs. Otherwise, this parameter is considered a non-key CSP.

² Must meet requirements of AS.06.05, AS.06.06 and AS.06.08 - These requirements cannot be enforced by administrative documentation and procedures but must be enforced by the cryptographic module itself.

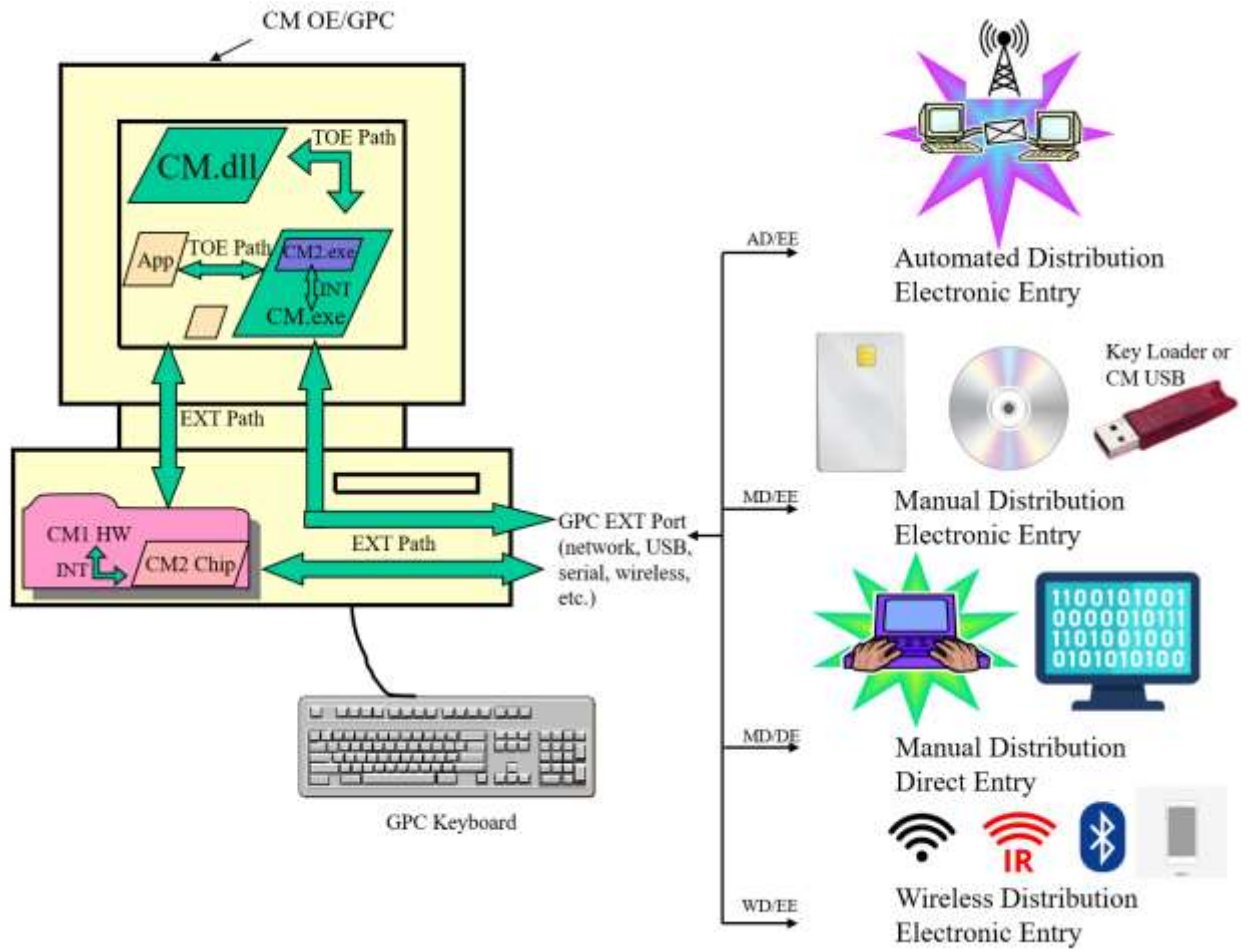
³ All AD / EE entries assume automated entry/output methods (without manual intervention or input) and therefore require an approved method listed in SP 800-140D. All other configurations assume manual entry (requiring human operator manipulation).

CM Software ² running on a networked GPC to/from EXT CM Hardware	AD / EE
CM Software ² to/from EXT CM Hardware or EXT CM Software via wireless connection	AD / EE
CM Hardware to/from EXT CM Hardware or EXT CM Software via wireless connection	AD / EE
CM Hardware to/from App Software with no user intervention via GPC EXT Path	AD / EE
CM Hardware to/from networked GPC	AD / EE
CM Hardware to/from EXT CM Hardware via EXT network port Path	AD / EE
CM Hardware (Sub-Chip Cryptographic Subsystem) to/from TOEPP CM Hardware (Sub-Chip Cryptographic Subsystem) via Single-Chip TOEPP Path at Levels 3 and 4	AD / EE
WD / EE – CM to/from CM via local Wireless⁴ EXT Path	
CM Software ² to/from CM Hardware or CM Software ² via local Wireless EXT Path	WD / EE
CM Hardware to/from CM Hardware via local Wireless EXT Path	WD / EE
N/A⁵ – CM to/from INT Path	
CM Software ² to/from CM Software ² via INT Path (CM embedded within CM)	N/A
CM Software ² hybrid to/from INT disjoint hardware component via INT Path	N/A
CM Hardware (Sub-Chip Cryptographic Subsystem) to/from TOEPP CM Hardware (Sub-Chip Cryptographic Subsystem) via Single-Chip TOEPP Path at Levels 1 and 2	N/A
CM Hardware to/from INT CM Hardware via INT Path (CM embedded within CM)	N/A

⁴ Wireless: wireless electronic entry or output per **ISO/IEC 24759:2017 AS09.18** is restricted to local wireless connections. These occur over short distances using short wavelength and lower power (e.g. Bluetooth, Infrared, Induction, Ultra-Wideband and other non-networking wireless technology).

⁵ N/A because SSP travels internal to the cryptographic boundary and does not enter/exit through the defined HMI, SFMI, HFMI or HSMI interfaces.

The following illustration provides reference to the above CSP Establishment table.



CSP Entry Format – Table 2

		Distribution (Establishment)											
		Manual				Automated				Manual Wireless			
Entry (Input / Output)	Direct	Keyboard, Number pad, Thumbwheel, Switch, Dial, etc.											
		1 ⁶	2 ⁶	3	4								
		P/E/SE	P/E/SE	TC/SK/E/SE	TC/SK/E/SE								
	Electronic	Smart Cards, Token, PC card, Diskettes, Key Loaders, OS, etc.				SSP Establishment SSP Transport or SSP Agreement				Bluetooth, Induction, Infrared (IR), Ultra Wideband (UWB), etc.			
		1 ⁶	2 ⁶	3	4	1	2	3	4	1	2	3	4
		P/E/SE	P/E/SE	TC/SK/E/SE	TC/SK/E/SE	SE	SE	SE	SE	E/SE	E/SE	E/SE	E/SE

Legend:

P⁷: Plaintext

E⁸: Encrypted using an approved security function listed in **SP 800-140C** (sections 6.2.2 or 6.2.6)

SE: SSP Establishment that uses an approved or allowed method listed in [IG D.F](#) or [IG D.G](#).

TC⁹: Trusted Channel per **ISO/IEC 19790:2012** Section 7.3.4 and [IG 3.4.A](#)

SK: Plaintext Split Knowledge using a TC

/ If items are separated by a “/”, the requirement can be met by *any* of the separated methods

Additional Comments

This IG reaffirms that SSPs established using *manual* transport methods and *electronically* or *directly* input or output to a cryptographic module may be input or output in plaintext at Security Levels 1 and 2.

Level 1 Software – General Purpose Operational Environment

AS06.05: (Level 1 and 2) Each instance of a cryptographic module shall have control over its own SSPs.

AS06.06: (Level 1 and 2) The operational environment shall provide the capability to separate individual application processes from each other in order to prevent uncontrolled access to CSPs and

⁶ Per **ISO/IEC 19790:2012** Section 7.9.5 (Security Levels 1 and 2): “For software modules or the software components of a hybrid software module, CSPs, key components and authentication data may be entered into or output in either encrypted or plaintext form provided that the CSPs, key components and authentication data **shall [09.19]** be maintained within the operational environment and meet the requirements of 7.6.3.”

⁷ Per **ISO/IEC 19790:2012** Section 7.9.5: “To prevent the inadvertent output of sensitive information, two independent internal actions **shall [09.16]** be required in order to output any plaintext CSP.”

⁸ Note: The algorithms used for option “E” **shall** be approved encryption/decryption algorithms, and other algorithms such as only using hash or keyed hash, are not permitted. Also, encryption methods are *not* required to be cryptographically authenticated for this option.

⁹CSPs that are *keys* **shall** not use this option (but can use the SK option to split the key while using a TC).

uncontrolled modifications of SSPs regardless if this data is in the process memory or stored on persistent storage within the operational environment.

AS06.08: (Level 1 and 2) Processes that are spawned by the cryptographic module shall be owned by the module and are not owned by external processes/operators.

A Software Cryptographic Module (SCM) requires the use of an underlying General-Purpose Computer (GPC) and Operational Environment (OE) to execute/operate. The OE of a software (or firmware or hybrid) module includes, at a minimum, the module components (e.g. the SCM), the computing platform (CP), and the operating system (OS) that controls or allows the execution of the software (or firmware) on the computing platform. The CP and OS of the OE which the software executes in are external to the defined SCM cryptographic boundary. The SCM executes/operates within the Tested OE's Physical Perimeter (TOEPP). The SCM is the collection of executable code that, at a minimum, encompass all security relevant algorithms, security functions, processes and components of a cryptographic module (e.g., dll's, exe's). Other general-purpose application software (App) (e.g., word processors, network interfaces, etc.) may reside within the TOEPP but outside of the SCM cryptographic boundary. Therefore, the TOEPP encompasses the following elements: GPC, OE, SCM and App. The SCM relies on the OE and GPC for memory management, access to ports and interfaces, and other services (which some of them are enforcing security requirements such as **AS06.05, AS06.06 and AS06.08**). The SCM has no operational control over other App elements within the TOEPP. The non-SCM elements (GPC, OE and App) are external to the cryptographic boundary of the SCM.

Example: If the SCM generates keys, it must use an approved DRBG. That key may be stored within the SCM without needing to follow guidance in Table 1 and Table 2. However, if the key is exported from the SCM cryptographic boundary, refer to Table 1 and Table 2 for the SSP establishment and key entry requirements. If a key is generated outside of the SCM, then the generation method is out-of-scope, but the key must be imported per Table 1 and Table 2 requirements.

It is the burden of the operator of the SCM to understand the environment the SCM is running on. If the operating system requirements of **AS06.05, AS06.06 and AS06.08** cannot be met, then the SCM cannot be validated (see additional requirements for Security Level 2 in **ISO/IEC 19790:2012** section 7.6.3). Restrictions to the configuration of the operational environment **shall** be documented in the Security Policy of the cryptographic module (**AS06.07**). **AS06.05, AS06.06, and AS06.08** requirements cannot be enforced by administrative documentation and procedures but must be enforced by the cryptographic module itself.

9.6.A Acceptable Algorithms for Protecting Stored SSPs

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	
Relevant Test Requirements:	
Relevant Vendor Requirements:	

Background

Rules for SSP storage are described in general terms in **ISO/IEC 19790**. The standard, however, does not list the specific approved or allowed methods for encrypting SSPs stored within the cryptographic module.

Question/Problem

In Section 7.9.6 of **ISO/IEC 19790:2012** it is stated that “SSPs stored within a module may be stored either in plaintext or encrypted form.” What does this mean? The preceding statement may appear to indicate that there are no requirements on SSP storage inside the module. However, the zeroization requirement does apply to “all unprotected SSPs and key components within the module.” SSPs that are protected are not plaintext and are

exempt from this requirement. An SSP can be *cryptographically* protected (e.g. encrypted), or a Public Security Parameter (PSP) can be *logically* protected if it cannot be modified or if its modification can be determined by the module. Therefore, it is necessary to know what constitutes, in this context, an acceptable *cryptographic* protection of stored SSPs.

Further, **ISO/IEC 19790:2012** Section 7.9.1 says, “Encrypted CSPs refer to CSPs that are encrypted using an approved security function. CSPs encrypted or obfuscated using non-approved security functions are considered unprotected plaintext within the scope of this International Standard.” Therefore, what is considered an “approved security function” in the context of SSP storage? In particular, it should be made clear whether the encryption of a stored SSP using a symmetric-key-encryption algorithm such as AES or the Triple-DES needs to satisfy the same requirements that apply to the protection of the cryptographic CSPs that are transported (via automated methods) in and out of the module. The latter requirements are described in [IG 9.5.A](#) and approved symmetric key transport methods are defined in **SP 800-38F**.

Resolution

SSPs may be stored within a module in any form – encrypted or unencrypted. To make a claim that SSPs are stored encrypted, or, more precisely, cryptographically “protected”, the module **shall** protect them using one of the following algorithms:

- An AES or a Triple-DES encryption using any approved mode of AES or the Triple-DES as defined in **SP 800-140C CMVP Approved Security Functions**.
- An RSA-based key encapsulation that may either comply with the requirements of **SP 800-56Brev2** or be allowed by [IG D.G](#).
- An approved hash algorithm for a CSP such as a password that does not need to be recovered but is used to check if it matches any other values. Approved hash algorithms are defined in **SP 800-140C**.

FIPS 140-3 also allows SSPs to be stored within another embedded validated module. Except at level 4, PSPs and CSPs physical or logically protected within such a FIPS 140-3 validated module are considered protected and are not required to be zeroized by this module.

The requirements of **SP 800-131A** for the encryption and key encapsulation key sizes apply if a stored SSP is claimed to be protected.

Additional Comments

1. Even though this guidance does not mandate the use of authenticated encryption algorithms from **SP 800-38F** it is *highly* recommended vendors adopt them because these algorithms are specifically designed to protect the confidentiality and the authenticity/integrity of cryptographic keys.
The approved algorithm implementations used to protect stored SSPs **shall** be tested by the CAVP (or vendor affirmed if allowed by an IG).

2. It follows from this IG and [IG D.G](#) that if the AES or the Triple-DES encryption is used then the requirements for encrypting stored SSPs are different from those when keys are transported in and out of the module. It is, however, strongly recommended that the rules of [IG D.G](#) are followed in this case as well.

If an RSA-based key encryption (encapsulation) is used to protect SSPs, the requirements are the same regardless of whether or not the protected SSP leaves the module’s boundary.

3. If the AES or the Triple-DES encryption is used to protect a stored SSP, the key encryption key may be established as shown in **SP 800-132**.
4. The **SP 800-131A** notation in this Implementation Guidance refers to the latest published revision of this standard.
5. As explained in the Background, a PSP is considered protected if it cannot be modified or if its modification can be determined by the module. A module may use non-cryptographic methods to determine whether a PSP has been modified.

9.7.A Zeroization of One Time Programmable (OTP) Memory

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.28, 09.29, AS09.30</i>
Relevant Test Requirements:	<i>TE09.28.01-06 TE09.29.01-02</i>
Relevant Vendor Requirements:	<i>VE09.28.01-06 VE09.29.01-02</i>

Background

AS09.28: (Sensitive security parameter zeroisation – Levels 1, 2, 3, and 4)

A module **shall** provide methods to zeroise all unprotected SSPs and key components within the module.

AS09.29: (Sensitive security parameter zeroisation — Levels 1, 2, 3, and 4)

A zeroised SSP **shall not** be retrievable or reusable.

AS09.30: (Sensitive security parameter zeroisation — Levels 2, 3, and 4)

The cryptographic module **shall** perform the zeroisation of unprotected SSPs (e.g. overwriting with all zeros or all ones or with random data).

The One Time Programmable (OTP) memory is a form of digital memory where the setting of each bit is locked by a fuse or antifuse. It provides a flexible, field-programmable alternative to Read Only Memory (ROM).

Question/Problem

If OTP memory is used within a module, how can the module meet FIPS 140-3 zeroisation requirements? Are there specific zeroisation requirements for OTP memory implementations?

Resolution

OTP memory can be used for storing plaintext secret, private or public cryptographic keys and SSPs within the module. However, the module **shall** be implemented in the following way in order to meet FIPS 140-3 zeroisation requirements:

1. Given that the OTP memory should be writable during module operation, the module **shall** provide the operator with the ability to zeroise all unprotected SSPs stored in OTP memory by overwriting the memory with 0s or 1s or random data. This will likely decommission the module but will prevent attackers from gaining knowledge of unprotected secret or public data stored within the OTP memory.
2. After the unprotected SSPs have been zeroised from the OTP memory, the module **shall** recognize the zeroised value as invalid and restrict the use to this value. This might be by using an additional bit that is flipped, or else code that knows the zeroisation value is invalid such as an integrity value that is not correct after zeroisation.
3. OTP memory is more likely than other types of data storage to have integrity values associated with the information. Therefore, any integrity value on the OTP memory **shall** be subject to zeroisation unless the vendor demonstrates that it could not leak information about the original key.

Additional Comment

A PSP is considered protected if it cannot be modified or if its modification can be determined by the module. A PSP stored with an integrity value so that the module can determine if it's been modified, is protected, and therefore, would not require zeroization.

Section 10 – Self-tests

10.3.A Cryptographic Algorithm Self-Test Requirements

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Transition End Dates	<i>December 31, 2020</i> – for select algorithms
Relevant Assertions:	<i>AS10.01-06, AS10.25-35</i>
Relevant Test Requirements:	<i>TE's associated with AS's above</i>
Relevant Vendor Requirements:	<i>VE's associated with AS's above</i>

Background

ISO/IEC 19790:2012 Section 7.10.1:

All self-tests **shall [10.01]** be performed, and determination of pass or fail **shall [10.02]** be made by the module, without external controls, externally provided input text vectors, expected output results, or operator intervention or whether the module will operate in an approved or non-approved mode.

Conditional self-tests **shall [10.04]** be performed when an applicable security function or process is invoked (i.e. security functions for which self-tests are required).

All self-tests identified in underlying algorithmic standards (Annexes C through E) **shall [10.05]** be implemented as applicable within the cryptographic module. All self-tests identified in addition or in lieu of those specified in the underlying algorithmic standards (Annexes C through E) **shall [10.06]** be implemented as referenced in Annexes C through E for each approved security function, SSP establishment method and authentication mechanism.

ISO/IEC 19790:2012 Section 7.10.3.1:

Conditional self-tests **shall [10.25]** be performed by a cryptographic module when the conditions specified for the following tests occur: Cryptographic Algorithm Self-Test, Pair-Wise Consistency Test, Software/Firmware Load Test, Manual Entry Test, Conditional Bypass Test and Conditional Critical Functions Test.

ISO/IEC 19790:2012 Section 7.10.3.2:

Cryptographic Algorithm Self-Test. A cryptographic algorithm test **shall [10.26]** be conducted for all cryptographic functions (e.g. security functions, SSP establishment methods and authentication) of each approved cryptographic algorithm implemented in the cryptographic module as referenced in [SP 800-140C through SP 800-140E]. The conditional test **shall [10.27]** be performed prior to the first operational use of the cryptographic algorithm.

A cryptographic algorithm self-test may be a *known-answer* test, a *comparison* test or a *fault-detection* test.

A *known-answer* test consists of a set of known input vectors (e.g. data, keying material, or constants in lieu of random bits) which are operated on by the cryptographic algorithm to generate a result. The result is compared to the known expected output result. If the calculated output does not equal the known answer, the cryptographic algorithm known answer self-test **shall [10.28]** fail.

An algorithm self-test **shall [10.29]** at a minimum use the smallest approved key length, modulus size, DSA prime, or curves as appropriate that is supported by the module.

If an algorithm specifies multiple modes (e.g. ECB, CBC, etc), at a minimum, one mode **shall [10.30]** be selected for the self-test that is supported by the module or as specified by the validation authority.

A *comparison test* compares the output of two or more independent cryptographic algorithm implementations, if the outputs are not equal, the cryptographic algorithm comparison self-test **shall [10.33]** fail.

A *fault-detection test* involves the implementation of fault detection mechanisms integrated within the cryptographic algorithm implementation, if a fault is detected, the cryptographic algorithm fault-detection self-test **shall [10.34]** fail.

ISO/IEC 19790:2012 Section 7.10.3.3:

If a cryptographic module generates public or private key pairs, a pair-wise consistency test **shall [10.35]** be performed for every generated public and private key pair as referenced in [SP 800-140C through SP 800-140E] for the applicable cryptographic algorithm.

Questions/Problems

What Cryptographic Algorithm Self-Tests (CASTs) are required for approved:

- symmetric-key algorithms (FIPS 197, SP 800-38 series, SP 800-67rev2)?
- hash algorithms that are not keyed (FIPS 180-4)?
- SHA-3 permutation-based and extendable-output functions (FIPS 202)?
- SHA-3 derived functions (SP 800-185)?
- hash algorithms that are keyed (FIPS 198-1)?
- deterministic random number generators (SP 800-90A)?
- entropy sources used for random bit generation (SP 800-90B)?
- key derivation functions, such as: KBKDF (SP 800-108), PBKDF (SP 800-132) and KDA (SP 800-56Crev1)?
- algorithms that are tested as a Component Validation List (CVL and [IG 2.4.B](#))?
- RSA, ECDSA, or DSA signature algorithms (FIPS 186-4)?
- key agreement schemes (SP 800-56Arev3)?
- asymmetric key transport schemes (SP 800-56Brev2)?
- vendor affirmed algorithms?

Please note that the requirements of this IG apply prior to the first operational use of the cryptographic algorithm.

Resolution

The ISO/IEC 19790 standard requires a CAST be conducted for all cryptographic functions of each approved cryptographic algorithm using, at a minimum, the smallest approved key length, modulus size, DSA prime, or curves as appropriate that is supported by the module; and if an algorithm specifies multiple modes (e.g. ECB, CBC, etc), at a minimum, one mode must be self-tested that is supported by the module or as specified by the validation authority. As the validation authority, this IG uses similar requirements as these but makes some changes as specified below:

- for symmetric-key algorithms, such as SKIPJACK, Triple-DES or AES,
 1. if the module implements an encryption function, this function **shall** be separately self-tested. If a known answer test (KAT) is used (rather than a *comparison* test or a *fault-detection* test), the module **shall** have an encrypted value pre-computed, perform the encryption using known data and key, and then compare the result to the pre-computed value;

2. if the module implements a decryption function, this function **shall** be separately self-tested. If a KAT is used (rather than a *comparison* test or a *fault-detection* test), the module **shall** have a decrypted value pre-computed, perform the decryption using known data and key (the data could be the encrypted value computed during the encryption test), and then compare the result to a value that was pre-computed value.
3. if the module implements the forward cipher function, then this forward cipher function **shall** be self-tested at least once by either performing a CAST on any encryption mode that supports the forward cipher function (typically all encryption modes support the forward cipher), or by selecting a decryption mode that supports the forward cipher function (e.g. CFB, OFB, CTR, CMAC, CCM, GCM). Similarly, if the inverse cipher function is implemented, then it **shall** be self-tested at least once by performing a CAST on any mode that supports the inverse function. Typically, the following modes support the inverse function within the decryption mode: ECB, CBC, XTS, KW, KWP.

The Encrypt CAST and Decrypt CAST do not need to be performed for each mode(s) that is selected to meet this cipher function requirement, as long the forward and inverse cipher functions (if implemented) are self-tested at least one time within any implemented mode(s). For example, if the module only implements AES GCM (Encrypt/Decrypt) and AES ECB (Encrypt/Decrypt), then the following CASTs would suffice: AES GCM Encrypt (to cover the forward cipher function), AES ECB Decrypt (to cover the inverse cipher function). CASTs on AES GCM Decrypt and AES ECB Encrypt would not be necessary. Or, in this example, the following CASTs would also suffice: AES ECB Encrypt (to cover the forward cipher function), AES ECB Decrypt (to cover the inverse cipher function). CASTs on AES GCM would not be necessary to meet this cipher function requirement.

4. Since symmetric modes that provide authentication (i.e. AES KW, KWP, GCM, CCM, CMAC, GMAC or Triple-DES CMAC and KW) are significantly more complex than those that perform only encryption/decryption (i.e. AES and Triple-DES ECB, CBC, OFB, CFB, CTR, and AES-XTS), a module **shall** perform a CAST on at least one authenticated encryption mode for each algorithm (i.e. AES and Triple-DES) if implemented by the module.

Moreover, some authenticated encryption modes are more complex than others (e.g. AES GCM requires a hash while AES KW only appends a known block to the plaintext to be encrypted as the authentication method). Therefore, below is the hierarchy to determine which mode(s) require a CAST, and which are covered by the higher-level self-test. If item (a) is self-tested, then this covers the requirements for items (b) and (c); if item (b) is self-tested, it covers items (c), and so on.

- (a) A CAST for one of the following AES authenticated encryption modes is required if implemented: GCM, CCM, CMAC, GMAC. A CAST for Triple-DES CMAC is required if this mode is implemented;
- (b) A CAST for AES-KW or KWP is required if no other AES authenticated encryption modes are implemented and self-tested; and a CAST for Triple-DES TKW is required if no other Triple-DES authenticated encryption modes are implemented and self-tested;
- (c) A CAST for a non-authenticated encryption mode (ECB, CBC, OFB, CFB, CTR, AES-XTS) is required if no other modes of the corresponding encryption algorithm (AES or the Triple-DES) are implemented and self-tested.

Please note that this hierarchy does not overrule the requirement above (bullet #3) to test at least one of each of the forward and inverse cipher functions (if implemented by the module), as that rule still applies regardless of which mode is selected to meet the authenticated encryption requirement. However, it is possible to use these rules in conjunction with each other (e.g. an encryption and decryption CAST on AES KW would cover the CAST requirements for all other non-authenticated AES modes implemented in a module, since AES KW is higher on the hierarchy list and also implements both the forward and inverse cipher functions).

Please also note that the above requirements to self-test a mode using separate encryption and decryption CASTs (bullets #1 and #2) apply only to modes that are required by this authentication encryption mode hierarchy. The encrypt/decrypt requirement does not apply to CASTs that are used to meet the cipher function requirement (bullet #3).

Note1: The SKIPJACK algorithm is an approved algorithm only for decryption so only a self-test for decryption is required.

- if the module implements a SHS function, the following **shall** be the minimal requirements for SHS algorithms
 - a CAST for SHA-1 is required;
 - a CAST for SHA-256 is required;
 - a CAST for SHA-224 is required if SHA-224 is implemented without SHA-256;
 - a CAST for SHA-512 is required;
 - a CAST for SHA-512/224 or SHA-512/256 is required if the SHA-512 CAST is not performed;
 - a CAST for SHA-384 is required if SHA-384 is implemented without SHA-512.
- if the module implements SHA-3 permutation-based and/or extendable-output functions (see [IG C.C](#) and **FIPS 202**):
 - At the minimum, the cryptographic module **shall** perform a CAST for one of the functions defined in **FIPS 202**: SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 and SHAKE256, no matter how many of these functions the module may be designed to use.
 - If a SHA-3 hash function is used as part of a higher-level approved algorithm that is tested by the CAVP, then the module **shall** perform a CAST for this higher-level algorithm. No additional CASTs for any of the **FIPS 202**-compliant functions implemented by this module are required.

Note1: The reason for requiring a self-test for only one of the **FIPS 202**-compliant hash functions is that all of these functions, including SHAKE128 and SHAKE256, rely on the same underlying Keccak-p permutation. Note that this is different from the SHA-1 and SHA-2 self-test requirements where separate self-tests are needed for SHA-1, SHA-256 and SHA-512, if the module is designed to use these hash functions.

Note2: If the module implements several Keccak-p permutation engines, a self-test **shall** be performed for more than one implementation of the **FIPS 202**-defined functions so that each permutation engine's implementation is self-tested.

- if the module implements SHA-3 derived functions (**SP 800-185**):
 - One CAST for one of the functions defined in **SP 800-185**: cSHAKE-128, cSHAKE-256, KMAC128, KMAC256, TupleHash and ParallelHash, is sufficient to meet the self-test requirements for all **SP 800-185** functions implemented in the module, as long as all use the same underlying SHAKE implementation.
 - If **FIPS 202** SHA-3 functions are in the same module and they share the same SHAKE implementation, then one CAST for either one **SP 800-185** function or one **FIPS 202** function is sufficient to meet the CAST requirement for **SP 800-185** functions.
 - If there are multiple SHAKE implementations, then each SHAKE implementation **shall** be self-tested.
 - If the module implements several Keccak-p permutation engines, a self-test **shall** be performed for more than one implementation of the **FIPS 202**-defined functions so that each permutation engine's implementation is self-tested.

Note1: The reason for requiring a CAST for only one of the **FIPS 202**-compliant hash functions is that all of these functions, including SHAKE128 and SHAKE256, rely on the same underlying Keccak-p permutation. Note that this is different from the SHA-1 and SHA-2 self-test requirements where separate self-tests are needed for SHA-1, SHA-256 and SHA-512, if the module is designed to use these hash functions

- if the module implements a HMAC function, a CAST for HMAC is required and **shall** be performed with the HMAC function using at least one of the implemented underlying SHS or SHA-3 algorithms.

- if the module implements an approved DRBG (**SP 800-90A**), then CASTs are required and **shall** be performed as specified in **SP 800-90A** Section 11.3. This requires separate known answer tests (KATs) for each implemented DRBG algorithm (e.g. separate tests for HMAC_DRBG, CTR_DRBG and HASH_DRBG) for the instantiate (11.3.2), generate (11.3.3), and reseed (11.3.4) functions of the DRBG.
 - The module may be optimized by combining the testing of Instantiate()-Generate() instead of testing each function separately. Similarly, the testing of Instantiate(), Reseed() and Generate() may be optimized in one sweep, since the new working state created by Reseed() is cryptographically chained to the state created by Instantiate(). A *known-answer* CAST of a DRBG may be performed by: Instantiate with known data, Reseed with other known data, Generate and then compare the result to a pre-computed value.
- if the module implements an approved ENT (**SP 800-90B**), then CASTs are required and **shall** be performed as specified in **SP 800-90B** Section 4.
- if the module implements an approved KBKDF (**SP 800-108**), the module **shall** perform a CAST for this algorithm covering at least one KDF option, even if the module supports multiple KBKDF options. That is, at least one mode (Counter, Feedback, or Double Pipeline) with at least one PRF **shall** be tested to cover a single KDF option.
- if the module implements an approved PBKDF (**SP 800-132**), the module **shall** perform a CAST, at minimum, on the derivation of the Master Key (MK) as specified in Section 5.3 of **SP 800-132**. In addition to performing a CAST on the MK derivation, the module **shall** self-test all underlying prerequisite algorithms implemented in the module that are used in the derivation of the Data Protection Key (DPK) key, if the same implementations of the underlying algorithms are not already self-tested either on their own or as part of other higher level algorithm self-tests. Therefore, the PBKDF CAST(s) **shall** at least cover a derivation of the Master Key (MK) and optionally DPK (if the underlying KDF and authenticated symmetric cryptography to derive the DPK are not tested separately). The easiest way to cover all PBKDF self-test requirements is to perform a CAST on the 2b option (if implemented) to test the derivation of the MK and the DPK, since deriving the DPK uses both the **SP 800-108** KDF and authenticated symmetric cryptography.
- if the module implements an approved KDA (**SP 800-56Crev1**), the module **shall** perform at least one CAST covering a **SP 800-56Crev1** Section 4 one-step KDF (if implemented) and another CAST covering a **SP 800-56Crev1** Section 5 two-step KDF (if implemented), including at least one auxiliary function for each. For example, if a module implements both a two-step KDF using either the HMAC or AES-CMAC auxiliary function, and one-step KDF using either the hash or HMAC auxiliary function, then at least two CASTs are required: one for the two-step KDF (with either HMAC or AES-CMAC), and the other for the one-step KDF (with either the hash or HMAC). The implementations of the auxiliary functions used in the **SP 800-56Crev1** CASTs do not require separate self-tests.

In addition, the module **shall** self-test all underlying prerequisite algorithms used in the remaining **SP 800-56Crev1** schemes implemented in the module, if the same implementations of the underlying algorithms are not already self-tested either on their own or as part of other higher-level algorithm self-tests.

Please note, all new module submissions after **December 31, 2020** **shall** comply with this requirement.

- if the module implements an approved CVL, this function is considered an algorithm “component” (and therefore part of a larger crypto function), and no CAST is required for the approved CVL. This is a general rule. However, there are cases, such in IGs [D.F](#) and [D.G](#), that require CASTs for CVL implementations.
- for each public key digital signature algorithm (RSA, DSA and ECDSA), a CAST **shall** be performed using at least one of the schemes approved for use in the approved mode. For example, if an RSA signature algorithm is self-tested using an X9.31-compliant scheme, it is not necessary to perform any additional CASTs for the implementations of the digital signature compliant with RSASSA-PSS or RSASSA-PKCS1-v1_5, even if these schemes are also supported by the module.
- for the RSA algorithm,

1. if the module implements digital signature generation, the module **shall** have an RSA digital signature CAST. If a KAT is used (rather than a *comparison* test or a *fault-detection* test), the RSA digital signature **shall** be pre-computed, generate an RSA digital signature using known data and key, and then compare the result to the pre-computed value;
2. if the module implements digital signature verification, the module **shall** have an RSA digital signature CAST. Similarly, if a KAT is used, the RSA digital signature **shall** be pre-computed (which could be the output of the RSA digital signature generate test), and using a known key, verify the signature by comparing the recovered message with its target value.
3. if the module does not implement RSA signature generation or RSA signature verification, but only implements RSA key encapsulation and un-encapsulation schemes for key transport described in **SP 800-56Br2** and [IG D.G.](#), the module is required to perform the **SP 800-56Br2** CASTs as described in [IG D.G.](#)

Note1: an RSA CAST **shall** be performed using both the public and private exponents (e and d) and the two exponents **shall** correspond [that is, $d * e = 1 \pmod{\text{LCM}(p-1, q-1)}$]. The public exponent e used in this RSA CAST **shall** be chosen from the public exponent values supported by the module.

Note2: an RSA CAST **shall** be performed at a minimum on any one approved modulus size and hash size (if applicable) that is supported by the module.

Note3: The CMVP will not validate RSA digital signature algorithms as approved in modules that implement a pair-wise consistency test in lieu of a CAST.

- for algorithms whose output vary for a given set of inputs such as DSA, ECDSA, and RSA PSS, they **shall** be self-tested as a CAST for signature generation or verification. For a KAT, this requires the randomization parameter be fixed. For a *comparison* test, this requires the randomization parameter be shared across all compared implementations.

Note1: a CAST **shall** be performed at a minimum on any one approved modulus or curve size that is supported by the module.

Note2: an ECDSA CAST **shall** be performed at a minimum, on any one of the implemented curves in each of the implemented two types of fields (i.e., prime field where $GF(p)$, and binary field where $GF(2^m)$).

- key agreement schemes (see [IG D.F.](#), **SP 800-56Arev3** and **SP 800-56Brev2**):
 - self-test requirements for approved key agreement schemes are shown in [IG D.F.](#)
 - all new module submissions after **December 31, 2020** **shall** comply with these self-test requirements.
- key transport schemes (see [IG D.G.](#) and **SP 800-56Brev2**):
 - self-test requirements for approved key agreement schemes are shown in [IG D.G.](#)
 - all new module submissions after **December 31, 2020** **shall** comply with these self-test requirements

If the module contains different implementations of a single algorithm, each algorithm implementation which can be executed simultaneously **shall** be self-tested separately.

All other approved algorithms that have been issued a CAVP certificate **shall** be self-tested unless an IG specifically reduces the requirement.

If an algorithm's implementation in the module is vendor-affirmed, then there is no self-test requirement unless an IG specifically requires it.

Prior to an algorithms first operational use, at least the minimum CAST that is required by this IG for that algorithm **shall** be performed, even if the CAST does not use the same key size, mode, modulus, etc. of the algorithm that is being used. For example, if a module implements both AES GCM and AES ECB, and a CAST for AES GCM is implemented to cover both the AES GCM and AES ECB self-test requirements, the AES GCM CAST **shall** be performed prior to the first use of either of these algorithms, not just before the AES GCM is used. In another example, if a module implements both RSA 2048 and 3072 Signature Verification functions, then, at minimum, either an RSA 2048, or RSA 3072 Signature Verification CAST **shall** be performed prior to the use of either of these functions.

Furthermore, it is not necessary to self-test all cryptographic functionality of an algorithm before it is used for only one purpose. For example, if a module implements both RSA 2048 Signature Generation and Signature Verification functions, then only an RSA 2048 Signature Verification CAST is required prior to the use of the Signature Verification functions. The RSA 2048 Signature Generation CAST must be performed prior to the use of the RSA Signature Generation function, but this can be done at a later time than the Signature Verification CAST.

Additional Comments

1. Though not a CAST, a pairwise consistency test (PCT) **shall** be conducted for every generated public and private key pair for the applicable approved algorithm (per **ISO/IEC 19790:2012** Section 7.10.3.3). To further clarify, at minimum, the PCT that is required by the underlying algorithm standard (e.g. **SP 800-56Arev3** or **SP 800-56Brev2**) **shall** be performed. If the underlying standard does not require a PCT on a key-pair (e.g. **SP 800-56Arev3** Section 5.6.2.1.4 option a.), then the appropriate PCT as specified in **TE10.35.01** (for key transport), **TE10.35.02** (for signatures), or **TE10.35.03** (for key agreement) **shall** be performed. The PCT **shall** be performed prior to the first operational use of either (private or public) key within an algorithm or scheme. The module **shall** know the purpose of the key pairs (for key transport, signatures or key agreement) either when generated/imported, prior to the first exportation, or prior to the first usage (if not exported before the first usage), and perform the corresponding PCTs.
2. Though not a CAST, continuous health tests **shall** be conducted on an approved ENT (**SP 800-90B**) as specified in **SP 800-90B** Section 4 (under **ISO/IEC 24759:2017 AS10.05**).
3. The reason the PBKDF CAST does not require self-testing the derivation of the DPK is because the derivation of the DPK relies entirely on either an approved **SP 800-108 KDF** and/or an approved authenticated encryption method, or neither, so as long as these are self-tested separately and are the same implementation, then repeating the self-test on these as part of the PBKDF CAST is not required.
4. Unless otherwise stated, all of the self-tests as mentioned in the Resolution in this IG can be met by either a *known answer test*, a *comparison test* or a *fault-detection test*.

10.3.B Self-test for Embedded Cryptographic Algorithms

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS10.04, AS10.05</i>
Relevant Test Requirements:	<i>TE10.25.01-02</i>
Relevant Vendor Requirements:	<i>VE10.25.01</i>

Background

Core cryptographic algorithms are often embedded into other higher cryptographic algorithms for their operation in an approved mode (e.g. SHA-256 algorithm embedded into HMAC-SHA-256 and ECDSA). However, when the cryptographic module performs a self-test on the higher cryptographic algorithm, the embedded core cryptographic algorithm may also be self-tested.

ISO/IEC 24759:2017

AS10.04: (Self-tests — Levels 1, 2, 3, and 4) Conditional self-tests shall be performed when an applicable security function or process is invoked (i.e. security functions for which self-tests are required).

AS10.05: (Self-tests — Levels 1, 2, 3, and 4) All self-tests identified in underlying algorithmic standards ({ISO/IEC 19790:2012} Annex C to Annex E) shall be implemented as applicable within the cryptographic module.

Question/Problem

If an embedded core cryptographic algorithm is self-tested during the higher cryptographic algorithm self-test, is it necessary for the cryptographic module to implement a self-test for the already self-tested core cryptographic algorithm implementation?

Resolution

It is acceptable for the cryptographic module not to perform a self-test on the embedded core cryptographic algorithm implementation if:

1. the higher cryptographic algorithm uses that implementation and,
2. the higher cryptographic algorithm performs a self-test prior to first operational use of the embedded algorithm (ISO/IEC section 7.10.3.1) and,
3. the higher cryptographic algorithm performs a self-test of the embedded algorithm as part of the on-demand initiation of periodic self-tests (ISO/IEC section 7.10.3.8) and,
4. all cryptographic functions within the embedded core cryptographic algorithm are self-tested (e.g. encryption and decryption for AES).

Additional Comments

1. If the cryptographic module contains several core cryptographic algorithm implementations (e.g., several different implementations of SHA-256 algorithm) and some are not used by other higher approved cryptographic algorithms (and are therefore not self-tested), then the cryptographic module must perform a separate self-test for each of those implementations.
2. Symmetric algorithm modes vary in complexity. [IG 10.3.A](#) requires that at least the most complex mode be self-tested. Self-testing of embedded encryption and decryption of an algorithm (e.g. AES) may not satisfy the entire self-test requirement for that algorithm.

Section 11 – Life-cycle assurance

Section 12 – Mitigation of other attacks

Annex A – Documentation requirements

Annex B – Cryptographic module security policy

Annex C – Approved security functions

C.A Use of non-Approved Elliptic Curves

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

The NIST **FIPS 186-4** standard allows (in Section 6.1.1) the use of non-NIST-Recommended (non-approved) curves in the ECDSA algorithm in the approved mode. However, **FIPS 186-5**, in its companion publication, **SP 800-186**, specifies the approved elliptic curves for ECDSA (including Deterministic ECDSA) and ECC KAS and does not include any provision for generating non-approved elliptic curve domain parameters. **SP 800-56A Rev3** states “The ECC domain parameters for key establishment for U.S. Government applications shall be selected only from the elliptic-curve domain parameters in **SP 800-186** that are listed in Appendix D, along with the security strengths that can be supported by each curve.” [IG D.F](#) Scenario 3 allows the use of non-approved methods (that is, not compliant with **SP 800-56A Rev3** and using non-NIST recommended curves) in the key agreement schemes in an approved mode of operation.

Question/Problem

Are there allowed elliptic curves for use in the ECDSA signature algorithm and the ECC-based key agreement schemes in an approved mode of operation? If so, what are the requirements for their use?

Resolution

Elliptic curves approved for use in ECDSA are specified in **SP 800-186**, as implemented in **FIPS 186-5**. Elliptic curves approved for use in ECC-based key agreement schemes are specified in Appendix D of **SP 800-56A Rev3**.

Allowed elliptic curves for use in ECDSA and ECC-based key agreement schemes (covered by [IG D.F](#), scenario 3), fall into one of two categories:

1. Well-known named elliptic curves listed in this IG. The allowed elliptic curves in this category are:
 - a. (From **SP 800-186** Appendix H.1) The curves specified in Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation [RFC 5639], which support a security strength of 112 bits or higher. In particular, this includes brainpoolP224r1, brainpoolP256r1, brainpoolP320r1, 2349 brainpoolP384r1, and brainpoolP512r1
2. During the transition period (see Additional Comments) between **FIPS 186-4** and **FIPS 186-5**, i.e., until **FIPS 186-4** is withdrawn, elliptic curve domain parameters that are generated according to **FIPS 186-4** Section 6.1.1.

The CMVP allows the use of these allowed elliptic curves in an approved mode of operation provided the following requirements are met:

1. the algorithm implementation **shall** use approved underlying algorithms, such as the message digests,
2. the Security Policy **shall** list all approved and non-approved curves that are implemented,
3. the Security Policy **shall** indicate the associated security strength for all non-approved curves that are implemented.
4. [Category 2 only] The vendor **shall** check that the curve is not singular; provide to the CMVP the information about the curve's underlying field (which **shall** be either of a prime order or of the order 2^m , where m is prime) and about the number of points on the curve; present the factorization of the number of points on the curve into a large prime n and a co-factor h as shown in **FIPS 186-4**; verify that h is within the limits established in Table 1 of Section 6.1.1 of **FIPS 186-4**; check that the curve is non-anomalous (the number of points on the curve is not equal to the size of the field), and that the MOV condition is met for all $B \leq 100$. See **ANS X9.62** or the ECC textbooks for details, and
5. if ECDSA or KAS (using elliptic curve cryptography) is listed on the certificate's approved line, the algorithm implementation **shall** have been CAVP tested and validated for at least one approved elliptic curve.

Additional Comments

1. This IG is written under the assumption that the domain parameter generation sections (that correspond to sections 6.1.1 in **FIPS 186-4**) will be removed in the published version of **SP 800-186**.
2. EdDSA in **FIPS 186-5** **shall** only be used with the specified curves in **FIPS 186-5** (Ed25519 and Ed448).
3. After **FIPS 186-5** is released, guidance will be issued describing this transition period that will address the status of these curves after the completion of the transition period and any impact that may have on validated module status.

C.B Validation Testing of Hash Algorithms and Higher Cryptographic Algorithm Using Hash Algorithms

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

The Cryptographic Algorithm Validation Program (CAVP) validates every approved hash algorithm implementation. Examples are: SHA-1 and SHA-2 (**FIPS 180-4**), and SHA-3 (**FIPS 202**). Several higher cryptographic algorithms use these hashing algorithms in their operation.

Question/Problem

What are validation testing requirements for the hash algorithms and higher cryptographic algorithms implementing hash algorithms for their use in an approved mode of operation?

Resolution

To be used in an approved mode of operation:

- every approved hash algorithm implementation **shall** be CAVP tested and validated on all of the module's operating environments.

- for higher level cryptographic algorithms that use these approved hash algorithms (e.g. RSA, ECDSA, KBKDF, HMAC, **SP 800-135** KDFs, **SP 800-56C** KDFs, etc.), every implemented combination for which CAVP testing exists, **shall**, upon the expiration of a transitional period defined when such CAVP testing becomes available, be CAVP tested and validated on all of the module's operating environments.
- where CAVP testing for an approved hash algorithm within a higher-level algorithm is NOT yet available or the transitional period has not yet expired, then for those combinations the vendor **shall** claim the vendor affirmation under relevant IGs (e.g. [IG C.C](#))

The algorithmic validation certificate annotates all the tested implementations that may be used in an approved mode of operation.

Any algorithm implementation incorporated within a FIPS 140-3 cryptographic module that is not either CAVP tested or vendor affirmed (if permitted by an IG) **shall** not be used in an approved mode of operation, unless provisions of **AS02.21** and [IG 2.4.A](#) are met. If there is an untested algorithm or subset of an approved algorithm, it would be listed as non-approved and non-compliant within the FIPS 140-3 validation's Security Policy.

C.C The Use and the Testing Requirements for the Family of Functions defined in FIPS 202

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

FIPS 202 was published in August 2015 and added to **SP 800-140C** in March 2020. This standard includes the specifications for the SHA-3 family of hash functions: SHA3-224, SHA3-256, SHA3-384 and SHA3-512, as well as for the two extendable-output functions, SHAKE128 and SHAKE256. CAVP testing for all of these functions became available on January 29, 2016.

Question/Problem

1. Are there any limitations on the use of hash functions defined in **FIPS 202** in the CMVP-validated cryptographic modules?
2. What are the validation testing requirements for the **FIPS 202**-compliant algorithms and the higher-level cryptographic algorithms that are using the functions defined in **FIPS 202**? In particular, how to address the case when CAVP testing is available for a higher-level cryptographic algorithm when this algorithm uses the hash functions defined in **FIPS 180-4** but there is still no testing when the same higher-level algorithm uses the **FIPS 202** functions?

Resolution

1. To be used in an approved mode of operation, the SHA-3 hash functions may be implemented either as part of an approved higher-level algorithm, for example, a digital signature algorithm, or as the standalone

functions. The SHAKE128 and SHAKE256 extendable-output functions may only be used as the standalone algorithms.

2. The validation, testing and the certificate documentation requirements are as follows.
 - a. Every implementation of each SHA-3 and SHAKE function **shall** be tested and validated on all of the module's operating environments.
 - b. If any of the SHA-3 hash functions are used as part of a higher-level algorithm and the CAVP testing that supports SHA-3 is available for this higher-level algorithm, then, upon the expiration of a transitional period defined when such CAVP testing becomes available, to use the higher-level algorithm in the approved mode the vendor **shall** obtain a CAVP certificate for this algorithm. If the vendor does not obtain such a certificate, then the higher-level algorithm that uses the SHA-3 functions may not be used in an approved mode and **shall** not be listed on the module's validation certificate.
 - c. If any of the SHA-3 hash functions are used as part of a higher-level approved algorithm and the CAVP testing that supports SHA-3 is NOT yet available for this higher-level algorithm or the transitional period has not yet expired, then to use this higher-level algorithm in the approved mode the vendor **shall** claim the vendor affirmation for this algorithm. This **shall** be accompanied by obtaining the CAVP certificates for the SHA-3 functions used in the higher-level algorithm. If the module implemented the same higher-level algorithm with a **FIPS 180-4** hash function and there is a corresponding entry on the approved line of the module's validation certificate, then the vendor affirmation of the same algorithm using SHA-3 does not need to be shown separately on the certificate's approved line but **shall** be documented in the module's Security Policy.
 - d. Until CAVP testing for a higher-level algorithm with the SHA-3 hash functions is available, the Management Manual Section 7.1 - *Addition of cryptographic security methods to SP 800-140C and SP 800-140D* is applicable.

Additional Comments

1. Examples for how to annotate the use of the **FIPS 202** algorithms in the module's validation certificate.
 - a. **SHA-3 (Cert. #A55)**. This demonstrates that one or more of the following functions: SHA3-224, SHA3-256, SHA3-384, SHA3-512 is implemented in the module and tested by the CAVP.
 - b. **SHAKE (Cert. #A55)**. This demonstrates that one or more of the following functions: SHAKE128, SHAKE256 is implemented in the module and tested by the CAVP.
 - c. **RSA (SHA-3 Cert. #A55, vendor affirmed)**. This is the case when the implementation of the RSA signature algorithm supported by the module uses only the SHA-3 hash functions, and no CAVP testing is available for the configuration of the RSA algorithm that uses the SHA-3 hash functions (or if such testing is available but the transitional period announced upon the introduction of this testing has not yet expired.)
 - d. **ECDSA (Cert. #A200)**. [No change from the existing notation.] The module's ECDSA algorithm implementation(s) may use both the **FIPS 180-4** and the **FIPS 202** hash functions. One entry in the module's validation certificate is sufficient. The Security Policy **shall** indicate that the use of the ECDSA with the SHA-3 hash functions is vendor affirmed if the CAVP testing for the ECDSA that uses the SHA-3 hash functions is not yet available.
2. The future updates of this Implementation Guidance will keep track of the testing availability status for the approved higher-level algorithms that might use the SHA-3 hash functions.

C.D Use of a Truncated HMAC

Applicable Levels:	All
--------------------	-----

Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01, TE02.20.02</i>
Relevant Vendor Requirements:	<i>VE02.20.01, TE02.20.02</i>

Background

The Keyed-Hash Message Authentication Code (HMAC) function is used by the message sender to produce a value, called the MAC, which is formed by condensing the secret key and the message input. The HMAC function may use any approved hash algorithm. HMAC is documented in [FIPS 198-1](#).

Some internet protocols such as IPsec and SSH use HMAC-SHA-1 and truncate the MAC to 96 (leftmost) bits. Other implementations use HMAC-SHA-384 truncated to 192 bits. Some other truncations may also be considered by implementers.

Question/Problem

Can a truncated HMAC be used in the approved mode?

Resolution

According to [SP 800-107rev1](#), published August 2012, the truncated forms of an approved HMAC are the approved algorithms if an HMAC output is truncated to its λ leftmost bits with $\lambda \geq 32$. In particular, HMAC-SHA-1-96 and HMAC-SHA-384-192 are approved algorithms and can be used in the approved mode of operation. This includes their use as an approved integrity technique required in Section 7.5 of FIPS 140-3 and as an approved authentication technique when performing the software/firmware load test described in Section 7.10.3.4 of FIPS 140-3.

Additional Comments

1. In compliance with the transition requirements specified in [SP 800-131A](#), any key for a full-length-output HMAC or a truncated HMAC that possesses less than 112 bits of strength **shall not** be used in approved mode.
 2. To be used in approved mode, the truncated HMAC **shall** receive a CAVP algorithm validation with the applicable truncated HMAC tested. The use of the truncated HMAC **shall** be shown in the module's Security Policy.
 3. The security of the truncated HMAC values is addressed in **SP 800-107rev1**. The permission to use in the approved mode an HMAC output truncated to (the minimum of) 32 bits does not contradict the requirement of **SP 800-131A** of providing at least 112 bits of equivalent encryption strength. The cryptographic strength of HMAC depends primarily on the strength of the HMAC key. See **SP 800-107rev1** for details.
 4. While **SP 800-107rev1** allows the truncation of HMAC to its λ leftmost bits with $\lambda \geq 32$, that Special Publication discourages the HMAC truncations to less than 64 bits. For the purposes of the FIPS 140-3 validations, it is the 32-bit requirement that will be enforced.
 5. HMAC-SHA-3 is subject to the same truncation rules as the other HMACs that utilize the approved hash functions.
 6. The **SP 800-131A** references in this Implementation Guidance refers to the latest published revision of this standard.
-

C.E Key Generation for RSA Signature Algorithm

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS07.16</i>
Relevant Test Requirements:	<i>TE07.16.01-02</i>
Relevant Vendor Requirements:	<i>VE07.16.01-02</i>

Background

SP 800-140C lists the approved security functions for FIPS 140-3. For asymmetric key digital signature standards, references address RSA signature generation, verification and key generation. Some of these referenced RSA standards include the specification of the RSA key generation procedure while others, such as RSASSA-PKCS1-v1_5 and RSASSA-PSS only define the requirements for signature generation and verification. These latter references do not address the generation of keys used in signature generation and verification.

Question/Problem

What methods for RSA key generation may be used when the module claims compliance with the RSA signature standards that do not explicitly address an RSA key generation method?

Resolution

If the module performs signature verification only, then the module does not need to possess a private RSA key and therefore does not need to generate it. The RSA public key parameters might be entered into the module or loaded at the time of manufacturing.

If the module performs an RSA Signature generation then the RSA private and public keys may either be loaded into the module (externally or pre-loaded at the time the module is manufactured) or generated by the module. If the module generates RSA signature keys then this key generation procedure **shall** be an approved method. The approved methods are described in **FIPS 186-5**. The module's RSA Signature CAVP algorithm certificate **shall** indicate that the RSA key generating algorithm has been tested and validated for conformance to the methods in **FIPS 186-5**.

Additional Comments

1. This Implementation Guidance will use **FIPS 186-5** and **FIPS 186-4** interchangeably during the outlined transition period for **FIPS 186-4**. When **FIPS 186-4** is no longer approved, this guidance will only refer to **FIPS 186-5**.
2. This Implementation Guidance does not address RSA key generation for use in the approved SSP establishment protocols. The user should follow the requirements of **SP 800-56B**.

C.F Approved Modulus Sizes for RSA Digital Signature for FIPS 186-4

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

The **FIPS 186-4** digital signature standard was published in July 2013. This standard specifies three possible RSA modulus sizes for signature generation and verification: 1024, 2048 and 3072 bits. Because of the transition to the stronger algorithms and key sizes, as documented in **SP 800-131A Rev2**, the 1024-bit RSA modulus may be used, for legacy purposes, in the approved mode for signature verification, but not for signature generation.

Question/Problem

SP 800-131A Rev2 provides only the lower bound, 2048 bits, for the RSA modulus size used in signature generation. Does this imply that the RSA modulus sizes other than 2048 and 3072 may be used to generate the RSA signatures in the approved mode? In particular, is the use of the 4096-bit modulus approved, and if so, what are the testing requirements for the RSA key generation if the key pair used in the RSA signature algorithm is generated by the module?

Resolution

When performing an RSA signature generation, a module may use any modulus size greater than or equal to 2048 bits. At least one of the RSA modulus lengths supported by the module for RSA signature generation **shall** be 2048, 3072, or 4096 bits. The RSA signature algorithm implementations **shall** be tested by a CST lab for all implemented RSA modulus lengths where CAVP testing is available. If there is no CAVP testing for the generation of RSA keys of a particular size, then the requirement of [IG C.E](#) to obtain a CAVP validation for the RSA key-generation algorithm does not apply to this key size.

Some of the RSA key generation methods described in Appendix B.3 of **FIPS 186-4** rely on the use of the auxiliary primes p_1 , p_2 , q_1 and q_2 that must be generated before the module generates the RSA primes p and q . Table B.1 in **FIPS 186-4** specifies, for RSA modulus lengths of 2048 and 3072 bits only, the minimum bit lengths and the maximum total length of the auxiliary primes. When implementing the RSA signature generation algorithm with other approved RSA modulus sizes, the vendor **shall** use the limitations from Table B.1 that apply to the longest RSA modulus shown in Table B.1 of **FIPS 186-4** whose length does not exceed that of the implementation's RSA modulus. The minimum number of the Miller-Rabin tests used in primality testing **shall** be consistent (based on the entries in Tables C.2 and C.3 in **FIPS 186-4**) with the bit sizes of p , q , p_1 , p_2 , q_1 and q_2 taken from Table B.1, as described above. Hence, for the 4096-bit RSA modulus, the bit size of each auxiliary prime p_1 , p_2 , q_1 and q_2 is greater than 170, and if the target prime-testing error probability is 2^{-100} , then the minimum number of the Miller-Rabin tests when generating and testing each of these primes is 27.

For example, when generating primes for the 4096-bit RSA modulus, the p and q primes **shall** be of 2048 bits each and the auxiliary primes **shall** be longer than 170 bits. The required minimum numbers of the Miller-Rabin tests are obtained from the bottom lines (those showing the 3072 modulus) of either Table C.2 or Table C.3 (for the target primality-testing error probability of 2^{-128} or 2^{-100} , respectively) of **FIPS 186-4**.

The use of the approved hash functions in digital signatures is documented in **SP 800-131A Rev2**, Table 8. The choice of a hash function may affect the security strength of the RSA signature algorithm.

Per [IG 10.3.A](#), an RSA CAST for signature generation may be implemented for any approved RSA modulus size supported by the cryptographic module; that is, any implemented RSA modulus size of at least 2048 bits.

When performing an RSA signature verification, a module may use a 1024-bit modulus, in addition to each RSA modulus size approved for use in signature generation.

Additional Comments

1. This Implementation Guidance is concerned with a description of the approved modulus sizes (and, therefore, the approved key sizes) for the RSA digital signature algorithm only. For completeness, here is the status of the approved and allowed key sizes in other asymmetric-key-based algorithms and schemes. This information is largely based on **SP 800-131A Rev2**.
 - a. RSA-based key transport (encapsulation) schemes. The modulus sizes of 2048 bits and larger are approved per **SP 800-56B Rev2**. All modulus sizes of 2048 bits and higher are allowed. It is strongly recommended that if an allowed scheme uses the auxiliary primes p_1 , p_2 , q_1 and q_2 , the limits on the

minimum sizes of these primes and on their maximum total size are the same as those stated in the main body of this IG for the auxiliary primes used in the RSA digital signature algorithm. The required numbers of the Miller-Rabin tests when testing the candidate primes are also the same as what is stated in the RSA digital signature case.

- b. DSA signatures. The approved bit sizes of the primes p and q used in the DSA algorithm are given in Section 4.2 of **FIPS 186-4**. Of these sizes, only the following pairs (2048, 224), (2048, 256) and (3072, 256) can be used in the approved mode for signature generation. For signature verification, these primes' sizes and the (1024, 160) pair are approved.
 - c. FFC-based key agreement schemes. The original publication of **SP 800-56A** showed the following three sets of the sizes of the p and q primes used in the FFC-based key agreement and shared secret computation schemes: FA: (1024, 160), FB: (2048, 224) and FC: (2048, 256). Of the three, only FB and FC are currently approved (see IG D.1-rev2 and IG D.1-rev3 for details), each resulting in the 112-bit strength in the established symmetric keys. In addition, any FFC-based scheme with the bit size of p no smaller than 2048 and the bit size of q no smaller than 224 is allowed in the approved mode.
 - d. ECDSA signatures. Digital signature generation and verification algorithms shown in **FIPS 186-4** are approved as long as the chosen elliptic curves are acceptable for their specific use. That is, all NIST-recommended curves are approved for signature verification. All but the following three NIST-recommended curves that provide less than 112 bits of encryption strength: P-192, B-163 and K-163, are also approved for signature generation. In addition, the use of any elliptic curve satisfying the requirements of [IG C.A](#) is allowed for signature verification. The use of these curves is also allowed for signature generation, if the bit length of n , as shown in Table 1 in **FIPS 186-4**, is least 224. It is vendor's responsibility to demonstrate that the non-approved curves meet these requirements.
 - e. ECC-based key agreement and shared secret computation schemes are approved, if they are compliant with one of the schemes specified in **SP 800-56A** (see [IG D.F](#) and **SP 800-140D** for approved **SP 800-56A** revisions) and the elliptic curve used would meet the requirements for the ECDSA signature generation as shown above. The elliptic curve used in a key-agreement scheme and the associated domain parameters **shall** provide at least 112 bits of security. See [IG D.F](#) for details.
 - f. RSA-based key agreement and DLC-based key transport schemes. These schemes are rarely used. There exist only the approved versions, no 'non-approved but allowed' ones. See the latest published revisions of **SP 800-56B** and **SP 800-56A** in Annex D for the detail explanations of these schemes and the security strength considerations.
2. When implementing a key agreement scheme (or a shared secret computation as part of a key agreement scheme), the vendor **shall** indicate in the module's Security Policy whether the scheme is of the Diffie-Hellman or the MQV variety. If a key agreement scheme (FFC or ECC-based) is documented on the module's certificate's non-approved line, the vendor is encouraged to state there if this is a Diffie-Hellman or an MQV scheme.
 3. A draft of **FIPS 186-5** was published in October 2019. It proposes several changes that may be addressed in a future IG.

C.G SP 800-67rev2 Limit on the Number of Encryptions with the Same Triple-DES Key

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>

Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

Background

[SP 800-67rev1](#) was published in January 2012. **SP 800-67rev1** added a requirement prohibiting users from performing more than 2^{32} 64-bit data block encryptions under the same three-key Triple-DES key. In an earlier version of **SP 800-67**, this requirement was a “should not” rather than a “**shall not**” statement. In compliance with **SP 800-67rev1**, NIST on July 11, 2017, placed on the Computer Security Division’s Website a document that explained the rationale for this restriction on the number of the Triple-DES encryptions using the same key.

Then, in November 2017, NIST published [SP 800-67rev2](#), which further tightened the restriction on the number of the Triple-DES encryptions with the same key. Per **SP 800-67rev2**, one key **shall not** be used to encrypt more than 2^{20} 64-bit data blocks. This version of the **SP 800-67** standard was included in the publication of **SP 800-140C** in March 2020.

Question/Problem

How **shall** the aforementioned evolving requirement on the limit of the number of the Triple-DES encryptions with the same key be enforced? In particular, how can a user of a validated cryptographic module be confident that other modules are not performing the Triple-DES encryptions with the same key thus, possibly, exceeding the overall limit on the number of such encryptions?

Resolution

Each validated module **shall** have a limit of either 2^{20} or 2^{16} 64-bit data block encryptions with the same Triple-DES key.

The limit of 2^{20} encryptions with the same Triple-DES key applies when keys are generated as part of one of the recognized IETF protocols. To use this provision, the Security Policy **shall** say which of the IETF protocols governs the generation of the Triple-DES keys and list the IETF RFC(s) where the details of this protocol, relevant to the generation of the Triple-DES encryption keys, are documented. A proof of the implementation’s compliance with the referenced protocol is not required.

The limit of 2^{20} encryptions with the same Triple-DES key also applies when the vendor can demonstrate the keys cannot be input, derived from inputs or shared secrets, or output, but may only be created within the module using its approved DRBG and used exclusively within the module. To use this provision, the Security Policy **shall** show how the key cannot be shared with any other instance of the module.

If a key is not generated as part of a recognized IETF protocol (or, at least, no such claim is made by the vendor) or not generated for only internal use within one module, then a further restriction on the number of encryptions with the same Triple-DES key is necessary, to avoid a “system-wide” violation of the overall limit on the number of encryptions with the same key. This limit is 2^{16} encryptions by each validated module.

The module itself **shall** enforce this requirement rather than enforcement by policy. See an Additional Comment #3 below for an example of how this *may* be done. The Security Policy **shall** explain how the module performs the enforcement.

Additional Comments

1. If an encryption key is generated as part of an IETF protocol implementation, there is a strong reason to believe (even though the module’s compliance to the protocol is not tested by the CMVP) that the same key will not be used by any entity except for the two parties that are involved in the encryption session that required the generation of this key. Therefore, if each module performs no more than 2^{20} encryptions with the same key, then system-wide the number of the Triple-DES encryptions with that key will usually be limited to 2^{20} or, *in the worst possible case*, if the module’s partner in this session can also perform the encryptions with the same key, to 2^{21} . While 2^{21} exceeds that stated limit, for the purposes of compliance with this IG, this solution is acceptable.
2. If an encryption key is not generated as part of a known protocol, or specific evidence of its limited use provided, then it is impossible to tell, in the general case, how many modules may use this encryption key.

The limit on the number of same-key Triple-DES encryptions is set at 2^{16} . If the number of modules using this key for encryption is no greater than 16 then the overall limit of 2^{20} will not be exceeded. In the highly unusual scenario when more than 16 modules share the same key, it is likely that at least some of these modules will not perform the number of encryptions that is close to the allocated maximum (2^{16}). Even if they did and the total number of same-key encryptions exceeded 2^{20} , it would be difficult for the attacker to increase the chances of success when the encryptions are performed by the unrelated modules.

3. Here is an example of how the module may enforce this requirement. The module will have a counter associated with each Triple-DES encryption key. When the counter reaches a certain value, the key can only be used for the decryption operations.

An easier solution would be for the module to have only one counter that will be increased by one every time the module performs a Triple-DES encryption. When the counter reading reaches the prescribed threshold, the module blocks all Triple-DES keys stored in the module at that time from being used to perform encryption. Of course, in this case the new Triple-DES encryption keys would need to be generated more often.

If the encryption counters are used in an implementation, then, in the case when the module's power is lost, the module may have a mechanism to restore the counters, or it may establish *all* new Triple-DES keys upon the restoration of power. The test report **shall** state which option, counter restoration or establishment of new Triple-DES keys the module uses for each Triple-DES key. If counter restoration is used, the test report **shall** explain how this mechanism guarantees that the counter value is restored to one no lower than the last value before loss of power.

4. The provisions of this IG apply to the Triple-DES key wrapping the same way as to the data encryption.
5. The provisions of this IG apply only to the three-key Triple-DES encryption. The use of the two-key Triple-DES encryption for the protection of sensitive data is not allowed.
6. As stated earlier in this IG, the module itself must enforce the limit on data block encryptions with the same Triple-DES key. In addition, **AS02.24** requires that the module services provide an indicator when utilizing an approved cryptographic algorithm in an approved manner. Therefore, the module can meet this AS for Triple-DES usage by either completely disabling the usage of a Triple-DES key once the limit is reached. Continuing to use the key while changing the service indicator from approved to non-approved violates **AS02.22**'s requirement that CSPs **shall** be exclusive between approved and non-approved services. If the module will permit a Triple-DES key to be used beyond the limit, then services **shall not** be marked approved while using that key, even prior to the limit being reached.

C.H Key/IV Pair Uniqueness Requirements from SP 800-38D

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

SP 800-38D was included in the publication of **SP 800-140C** in March 2020. **SP 800-38D** requires that “the probability that the authenticated encryption function ever will be invoked with the same IV and the same key on two (or more) distinct sets of input data **shall** be no greater than 2^{-32} ”.

One difficulty of testing the module's compliance with this requirement comes from the fact that each module is tested independently while **SP 800-38D** demands that the probability of the (Key, IV) pair collision between all modules at all times should be sufficiently low to ensure cryptographic strength.

Question/Problem

How **shall** a cryptographic module satisfy these requirements?

Resolution

An AES-GCM key may either be generated internally or entered into the cryptographic module.

Five techniques for generating an IV that are acceptable for the purposes of FIPS 140-3 validation are listed below.

1. Construct the IV in compliance with the provisions of a peer-to-peer industry standard protocol whose mechanism for generating the IVs for AES-GCM has been reviewed and deemed acceptable by the appropriate validation authorities and subject to the additional requirements established in this guidance. The current list of acceptable protocols is shown below:
 - a. TLS 1.2 GCM Cipher Suites for TLS, as described in RFCs [5116](#), [5246](#), [5288](#), and [5289](#) provisions;
 - b. IPsec-v3 protocol, as described in RFCs [4106](#), [5282](#), and [7296](#).
 - c. MACsec with GCM-AES-128, GCM-AES-256, GCM-AES-XPB-128 and GCM-AES-XPB-256 Cipher Suites, as described in [IEEE 802.1AE](#) (MACsec) and its amendments.
 - d. SSHv2 protocol defined in RFCs [4251](#), [4252](#), [4253](#), [4254](#) and [5647](#).

The following are additional specific requirements for each acceptable protocol for the purposes of FIPS 140-3 validation.

TLS 1.2 protocol IV generation

If an IV is constructed according to the TLS 1.2 protocol, then this IV may only be used in the context of the AES-GCM mode encryption within the TLS 1.2 protocol.

If the vendor claims that the IV generation is in compliance with the TLS 1.2 specification and only for use within the TLS 1.2 protocol, then the module's Security Policy and the Validation Test Report **shall** explicitly state the module's compatibility with TLS 1.2 and the module's support for acceptable AES-GCM ciphersuites from Section 3.3.1 of **SP 800-52 Rev 1** or **SP 800-52 Rev 2**.

For the purposes of this Guidance, the module may demonstrate its compliance with the rules for TLS 1.2 compliance in one of the following two ways.

- i) The operations of one of the two parties involved in the TLS 1.2 SSP establishment scheme **shall** be performed *entirely within* the cryptographic boundary of the module being validated. The testing laboratory **shall** check the module implementation and verify that the keys for the client and server negotiated in the handshake process (client_write_key and server_write_key) are compared and the module aborts the session if the key values are identical; or
- ii) The laboratory **shall** check the TLS 1.2 protocol implementation that relies on the module being validated against an independently developed instance of TLS 1.2, such as the many TLS 1.2 client test sites on the Internet, verify that a session is successfully established, which implies that the client_write_key and server_write_key values are derived correctly, and the following condition **shall** be met:

The module's implementation of AES-GCM is used together with an application that runs either inside or outside the module's cryptographic boundary. This application negotiates the protocol session's keys and the 32-bit nonce value of the IV. The nonce is positioned where there is the "name" field in Scenario 3 of this Guidance.

Whether an implementation is using the i) or the ii) path to meet the compliance requirements, the counter portion of the IV **shall** be set by the module *within* its cryptographic boundary and the requirements of Scenario 3 of this Guidance for the counter field (including the IV restoration conditions) are satisfied.

The implementation of the nonce_explicit management logic inside the module **shall** ensure that when the nonce_explicit part of the IV exhausts the maximum number of possible values for a given session key (e.g., a 64-bit counter starting from 0 and increasing, when it reaches the maximum value of $2^{64} - 1$), either party (the client or the server) that encounters this condition triggers a handshake to establish a new encryption key – see Sections 7.4.1.1 and 7.4.1.2 in RFC [5246](#). A statement to that effect **shall** be included in the Security Policy and Validation Test Report.

IPsec-v3 protocol IV generation

If an IV is constructed in compliance with the IPsec-v3 protocol, then this IV may only be used in the context of the AES-GCM mode encryption within the IPsec-v3 protocol.

If the vendor claims that the IV generation is in compliance with the IPsec-v3 specification and only for use within the IPsec-v3 protocol then the module's Security Policy and the Validation Test Report **shall** explicitly state the module's compliance with RFC [4106](#) and/or RFC [5282](#) (depending on the protocols supporting GCM). The Security Policy and Validation Test Report **shall** also state that the module uses RFC [7296](#) compliant IKEv2 to establish the shared secret SKEYSEED from which the AES-GCM encryption keys are derived.

Similar to the allowances shown above for the TLS 1.2 implementations, the module may demonstrate its compliance with the rules for IPsec-v3 compliance in one of the following two ways.

- i) The operations of one of the two parties involved in the IKEv2 SSP establishment scheme **shall** be performed entirely within the cryptographic boundary of the module being validated. The testing laboratory **shall** check the module implementation and verify that the two keys established by IKEv2 for one security association (one key for encryption in each direction between the parties) are not identical and abort the session if they are; or
- ii) The laboratory **shall** check the IPsec-v3 protocol implementation that relies on the module being validated against an independently developed instance of IPsec-v3 with IKEv2, verify that a session is successfully established, which implies that the two keys established by IKEv2 are derived correctly, and the following condition **shall** be met:

The module's implementation of AES-GCM is used together with an application that runs either inside or outside the module's cryptographic boundary. This application negotiates the protocol session's keys and the value in the first 32 bits of the nonce (see below).

Note that in RFC [5282](#) the term for what is called an IV in **SP 800-38D** and in this IG is "nonce", while the term "IV" in RFC [5282](#) refers only to the last 64 bits of the "nonce" field. In other words, IPsec-v3 requires four octets of salt followed by eight octets of deterministic nonce. Whether an implementation is using the i) or the ii) path to meet the compliance requirements, the construction of the last 64 bits of the "nonce" (the IV in RFC [5282](#)) for the purposes of FIPS 140-3 validation **shall** be deterministic (e.g., using a counter) and satisfy one of the IV restoration conditions defined in Scenario 3 of this Implementation Guidance.

The implementation of the management logic for the last 64 bits of the "nonce" (the IV in RFC [5282](#)) inside the module **shall** ensure that when the IV in RFC [5282](#) exhausts the maximum number of possible values for a given security association (e.g., a 64-bit counter starting from 0 and increasing, when it reaches the maximum value of $2^{64} - 1$), either party to the security association that encounters this condition triggers a rekeying with IKEv2 to establish a new encryption key for the security association – see RFC [7296](#). A statement to that effect **shall** be included in the Security Policy and Validation Test Report.

MACsec protocol IV generation

A typical implementation of the MACsec protocol includes the components shown in Figure 1 below. The generation and management of IV's for the GCM cipher suites in the MACsec protocol is distributed among the three components. For the purposes of a FIPS 140-3 validation, the cryptographic functionality relevant for MACsec in each component **shall** be validated as a separate module. The requirement in Section 9.1 in **SP 800-38D** to contain the IV "generation unit" within a module boundary is satisfied by the composition of the **Peer**, **Authenticator** and optional **Authentication Server** modules. All modules – **Peer**, **Authenticator**, and, if applicable, **Authentication Server**, should be validated (or re-validated) after this Implementation Guidance takes effect, so that they all comply with the applicable requirements of this IG. While all modules are validated separately by the CMVP, each module's Security Policy **shall** tell what this module's role is in the MACsec protocol, explain what the module does in support of the IV generation for the MACsec's use of AES-GCM, and state that when supporting the MACsec protocol in the approved mode, the module should only be used together with the CMVP-validated modules providing the remaining <**Peer**, **Authenticator**, ...> functionalities. The module **shall** satisfy one of the IV restoration conditions defined in Scenario 3 of this Implementation Guidance.

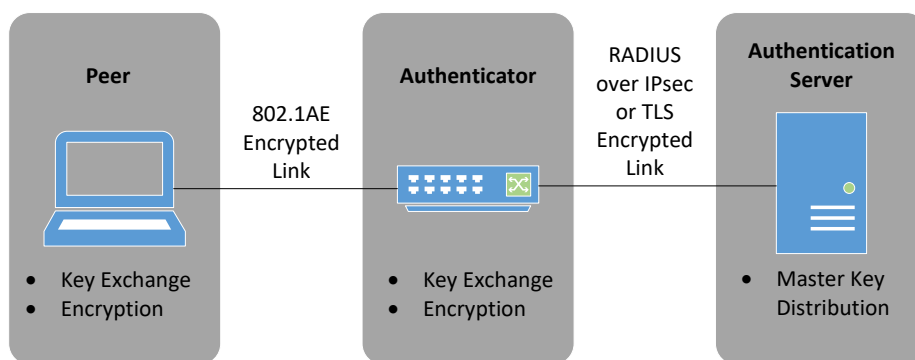


Figure 1. MACsec protocol components

The link between the **Authenticator** and the optional **Authentication Server** is typically implemented by the RADIUS protocol, which is a de-facto industry standard for communication to authentication servers. All references to RADIUS in this Guidance are applicable only to the use of this protocol in the MACsec context. To provide security the RADIUS traffic should be tunneled over an IPsec (cf. RFC [3162](#)) or TLS channel (cf. RFC [6614](#)). All configuration instructions for the link between the **Authenticator** and the **Authentication Server** **shall** be provided in the Security Policy of the module.

The **Peer** and the **Authenticator** Modules Security Policies **shall** state that the link between the **Peer** and the **Authenticator** should be secured to prevent the possibility for an attacker to introduce foreign equipment into the local area network – see Section 7.3 in IEEE Std 802.1X-2010.

SSHv2 protocol IV generation

A. Properties of the SSHv2 protocol

The current version of the SSH protocol is SSHv2. This protocol is defined in the IETF RFCs [4251](#), [4252](#), [4253](#) and [4254](#). The rules for using AES-GCM are documented in RFC [5647](#). One of the advantages of the SSHv2 protocol, for the purpose of meeting the (key, IV) probability collision requirements of **SP 800-38D**, is that the encryption and the message authentication parameters are negotiated separately for each direction, with independent keys. This can be seen in Figure 1 in RFC [5647](#) showing the derivation methods for sessions' keys. A shared secret K is negotiated as shown in Section 8 of RFC [4253](#), employing a Diffie-Hellman or an EC Diffie-Hellman scheme, and this shared secret is used to negotiate the independent encryption keys. This construction guarantees that a key collision between the encryption keys from separate sessions may occur only if there is a shared

secret collision or a hash collision. The probability of this happening is negligibly small in comparison with **SP 800-38D** bound of 2^{-32} for the tolerable probability of the (key, IV) pair collisions. Hence one should only be concerned with the probability of the IV collisions among the separate encryptions within the same SSHv2 session.

B. IV generation method

A new IV parameter is generated by the module for each AES-GCM encryption. As shown in Section 7.1 of RFC [5647](#), the IV consists of a 4-byte fixed field and an 8-byte invocation counter. The initial IV value is generated as shown in Figure 1 of RFC [5647](#) and the fixed field of this IV remains the same for the duration of the session. Therefore, the method for minimizing the (key, IV) collision probability within the same session depends entirely on the management of the invocation field of the IV. The invocation counter is treated as a 64-bit integer and is incremented by one when performing an AES-GCM encryption of a new binary packet. The formation of binary packets is explained in Section 7.2 of RFC [5647](#).

C. A Method of Compliance

If an IV is constructed in compliance with the SSHv2 protocol, then this IV **shall** only be used in the context of the AES-GCM mode encryptions within the SSHv2 protocol.

If the vendor claims that the IV generation is in compliance with the SSHv2 specification and only for use within the SSHv2 protocol, then the module's Security Policy and the Validation Test Report **shall** explicitly state the module's compliance with RFCs [4252](#), [4253](#) and [5647](#). The tester **shall** verify that the module's management of the invocation counter of the AES-GCM IV satisfies the following conditions:

- If the invocation counter reaches its maximum value $2^{64} - 1$, the next AES-GCM encryption is performed with the invocation counter set to either 0 or 1.
- No more than $2^{64} - 1$ AES-GCM encryptions may be performed in the same session. (This requirement may be met by either implementing an encryption counter or by reviewing the maximum number of encryptions that the module can produce.)
- When a session is terminated for any reason, a new key and a new initial IV **shall** be derived. (The session_id parameter in Figure 1 of RFC [5647](#) may remain unchanged.)

Meeting all of the requirements in this section will ensure that the **SP 800-38D** probability bound for the (key, IV) collisions will not be exceeded. The module **shall** satisfy one of the IV restoration conditions defined in Scenario 3 of this Implementation Guidance.

2. The IV may be generated internally at its entirety *randomly*. In this case,

- The generation **shall** use an Approved DRBG and
- The DRBG seed **shall** be generated inside the module's physical boundary.
- The IV length **shall** be at least 96 bits (per **SP 800-38D**). See Additional Comments for the discussion of why an IV of less than 96 bits may not be generated randomly and concatenated to the remaining part of an IV.
- The module's entropy-producing mechanism **shall** comply with the requirements of [IG D.J](#) and guarantee that at least 96 bits of entropy strength is generated.

A statement to that effect **shall** be included in the Security Policy and Validation Test Report.

3. If an AES-GCM Key is generated either internally or externally and the IV is constructed at its entirety internally *deterministically* then the requirement of **SP 800-38D** quoted in the Background section above will be modified.

Instead of requiring that the probability of any (key, IV) collision anywhere in the Universe at all times did not exceed 2^{-32} , it will only be required that for a given key distributed to one or more cryptographic modules, the (key, IV) collision probability would not exceed 2^{-32} . This is equivalent to

the requirement that for any key distributed to one or more modules the probability of a collision between the deterministically-generated IVs is no greater than 2^{-32} .

The module **shall** use at least 32 bits of the IV field as a name and use at least 32 bits as a deterministic non-repetitive counter for a combined IV length between 64 bits and 128 bits. The name field **shall** include an encoding of the module name and the name construction **shall** allow for at least different names. For example, if the module name is such that it consists of at least 8 hexadecimal characters then this condition is satisfied, since 16^8 is no smaller than (indeed, equal to) 2^{32} .

Alternatively, if the name consists of at least 6 alphanumeric characters, each having at least 62 values, then this is also sufficient. Even though not all possible names are equally likely to be used, the fact that the modules can possibly have at least 2^{32} different names will be sufficient to meet this requirement.

The implementation of the deterministic non-repetitive counter management logic inside the module **shall** ensure that when the counter part of the IV exhausts the maximum number of possible values for a given session key (e.g., a 32-bit counter starting from 0 and increasing, when it reaches the maximum value of $2^{32} - 1$) the encryptor **shall** abort the session.

Further, at least one of the IV restoration conditions **shall** be satisfied for the deterministic non-repetitive counter.

The IV restoration conditions are as follows (for additional details, see Section 9.1 of **SP 800-38D**):

- The module's memory **shall** be set in such a way that it will reset to the last IV value used in case the module's power is lost and then restored. (This condition is enforced by the module and **shall** be tested by a testing lab.)
- There will be a human operator who will reset the IV to the last one used in case the module's power is lost and then restored. (This condition is not enforced but **shall** be stated in the module's Security Policy, under the "User Guide" heading.)
- In case the module's power is lost and then restored, a new key for use with the AES-GCM encryption/decryption **shall** be established. (This condition may or may not be enforced but **shall** be stated in the module's Security Policy, under the "User Guide" heading.)

A statement explaining how the deterministic IV generation is performed and how the IV restoration conditions are met **shall** be included in the Security Policy and Validation Test Report.

NOTE. The module does not need to comply with any particular Scenario (1 – 3) shown in this section of the IG. Meeting the rules of any one of these Scenarios, whether protocol-dependent or not, when generating the IVs for AES-GCM is sufficient. The Security Policy **shall** explain the rules under which the module must operate in compliance with this Implementation Guidance.

4. If an implementation does not meet the requirements of any of the Scenarios 1 through 3 explained above in this Implementation Guidance, the vendor may present their own proof of the compliance with the **SP 800-38D** requirement stated in the **Background** section of this IG. The burden of proof is on the vendor. The testing laboratory **shall** review the proof and verify its correctness. Each proof will be examined by the CMVP reviewers who will make the final determination of the proof's validity.
5. If an implementation is generating an IV in compliance with the provisions of an industry¹⁰ protocol supporting AES-GCM encryption, that is not included among the acceptable protocols in Scenario 1 above, the vendor's may build their own case for the security of the IV generation method using the following guidelines:
 - The generated IV **shall** only be used in the context of the AES-GCM encryption executing the provisions of the protocol within which the IV was generated.

¹⁰ An "industry protocol" is either defined or documented and supported by one of the well-recognized standards bodies, such as the IEEE, IETF, ANSI or ISO SC27 (the list is not exclusive).

- The module's Security Policy **shall** state the protocol's name and version number and confirm that the IV is generated and used within this protocol's implementation.
- The Security Policy **shall** list the documents (such as the IETF RFCs) where the protocol and, specifically, the use of the AES-GCM encryption within the protocol are defined.

It is often the case that while a protocol's implementation may be compliant with **SP 800-38D** this implementation is distributed across several different modules and apps. Testing one cryptographic module at a time may not guarantee compliance with **SP 800-38D**; in particular, with the provision of this standard requiring that the (key, IV) pair collision probability does not exceed 2^{-32} . The vendor **shall** identify the features of the protocol and of the specific implementation, as well as the selected testing mechanisms and use this information to prove that the system that includes the module under test meets the **SP 800-38D** collision probability requirement, even if not all relevant operations are performed within one cryptographic module. The proof will be reviewed by the CMVP who will have the final word as for the correctness of the vendor's analysis.

Some of the following considerations may be used, when applicable, while constructing a proof. This list is not exclusive; depending on the protocol, the vendor and the testing lab may identify some other properties of the protocol or of the specific implementation that could be used to make a claim of the implementation's security.

- The protocol's implementation is contained within the boundary of the module under test.
- The protocol implementation may be split between several other modules and applications, but the AES GCM IVs are generated within the boundary of the module under test.
- The protocol implementation may be split between several modules and applications, each of them validated by the CMVP, thus providing the assurances of the overall compliance with the protocol's method for generating the AES GCM IVs.
- The vendor is running a protocol implementation that includes the module under test against an independently developed instance of the same protocol.
- The protocol properties might include the guarantees – with a very high probability - of the systemwide uniqueness of the key, which would reduce the (key, IV) collisions to the collisions of the IVs. The latter can be estimated, under the reasonable assumptions acceptable to the CMVP reviewers, by examining the IV construction process under the rules of the protocol.
- If portions of the IV value are generated within different modules/entities, the vendor's proof **shall** state which party generates which bits of the IV and explain why this method keeps the (key, IV) collision probability below the **SP 800-38D** bound.

Additional Comments

1. This Implementation Guidance does not introduce any new requirements. On the contrary, the purpose of this Implementation Guidance is to relax some of the **SP 800-38D** requirements. This relaxation is needed because **SP 800-38D** imposes some system-wide requirements, including those that concern the probabilities of the (key, IV) pair collisions. The compliance with such system-wide requirements cannot be tested within the scope of the CMVP program where each module is validated separately.

In some Scenarios allowed by this IG, the (key, IV) collision probability is calculated as it applies to one module. To reconcile this interpretation of the **SP 800-38D** requirement with the security goals stated in **SP 800-38D**, the module needs to operate within the certain limits established by this Implementation Guidance. One possibility is for the module to comply with the specific confines of the known industry protocols. If this is the case, the reliance on the properties of the protocol allows the CMVP to modify the rules of **SP 800-38D** without introducing any security risks or exposures.

Scenarios 2 and 3 in the **Resolution** section are protocol independent. Scenario 4 is the general case which permits the vendor to show the module's implementation's compliance with **SP 800-38D**. Scenario

5 allows the vendor to extend the protocol-specific cases of Scenario 1 developed by the CMVP to the protocols that are not examined in this Implementation Guidance.

2. When the module uses a (non-protocol-specific) deterministic IV construction, the name field provides an assurance that the IVs are not repeated even when they are generated independently by different modules, possibly manufactured by different vendors. If this name field is only 32 bits long, then it is barely sufficient to provide for the 2^{32} different values. Operators of the modules that use the 32-bit long names need to ensure that there is no possibility of name collisions in the systems they operate. When it is not possible to control the name assignment (as in the case when a session that uses an AES GCM encryption runs over a network managed without a central control over the names of the modules) then a 32-bit field may be insufficient. In such cases, it is highly recommended to switch to 64-bit or even 96-bit long names and limit the number of encrypted blocks under the same key to 2^{32} .
3. As stated in this Implementation Guidance, if the entire IV is generated randomly, the length of the random field **shall** be at least 96 bits. The reason for it is that with 2^{32} possible AES GCM encryptions under the same key, the probability of having at least one (key, IV) collision (i.e., an IV collision, as the key stays the same) can be estimated to be of the order of 2^{-32} . However, to maintain the same probability of collisions in the case of a 64-bit random field, one would have to reduce the maximum number of AES GCM encryptions with the same key to only 2^{16} .
4. Including the module's name in the IV field does not amount to a passphrase-based key derivation. The IV is not a key. Their cryptographic properties are different.
5. The methods presented in this IG apply to the module's generation of an IV parameter for the AES GCM encryption. When an IV is used for decryption, the responsibility for the IV generation lies with the party that performs the AES GCM encryption therefore none of the **SP 800-38D** requirements and none of the Scenarios presented in this Guidance are applicable to the module performing the decryption.
6. Some proprietary implementations of MACsec allow the static configuration of a pre-shared key for the Security Association Key (SAK) used by the protocol. The static configuration of a pre-shared SAK **shall not** be used in the approved mode of operation.
7. If any of the IETF or IEEE documents referenced in Scenario 1 of this IG become obsolete or get updated by another IETF RFC or an IEEE standard, then the new document number **shall** be considered the replacement of the number listed in this Guidance. However, a new *version* of a protocol (say, TLS 1.3) is not automatically covered by this Guidance. Until this new version of a protocol is included in Scenario 1 by the CMVP vendors may use Scenario 5 to demonstrate the required security properties of their modules' protocol implementations.

C.I XTS-AES Key Generation Requirements

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

XTS-AES is approved in **SP 800-38E** by reference to **IEEE Std. 1619-2007**. The IEEE standard specifies a key, denoted by *Key*, that is 256 [or 512] bits long; *Key* is then parsed as the concatenation of two AES keys, denoted by *Key_1* and *Key_2*, that are 128 [or 256] bits long. Sec. D.4.3 (pp. 31-32) explains that XTS-AES differs from the generic XEX construction (due to Rogaway) in that *Key_1* = *Key_2* for XEX but not for XTS-

AES. Annex D of the IEEE standard is labeled as informative, not normative and there are no other requirements on the generation of Key otherwise in that standard.

Question/Problem

Misuse of XTS-AES with a class of improper keys results in a security vulnerability. An implementation of XTS-AES that improperly generates *Key* so that *Key_1* = *Key_2* is vulnerable to a chosen ciphertext attack that would defeat the main security assurances that XTS-AES was designed to provide. In particular, by obtaining the decryption of only one chosen ciphertext block in a given data sector, an adversary who does not know the key may be able to manipulate the ciphertext in that sector so that one or more plaintext blocks change to any desired value. Rogaway illustrates the attack for disallowed parameterizations of XEX (without fully exploring its consequences) in Sec. 6 of his 2004 paper Efficient Instantiations of Tweakable Block ciphers and Refinements to Modes OCB and PMAC, available at <http://web.cs.ucdavis.edu/~rogaway/papers/offsets.pdf>.

Resolution

Key **shall** be generated to comply with the approved key generation guidelines of NIST SP 800-133 Rev. 2, Sec. 6.3, “Symmetric Keys Produced by Combining Multiple Keys and Other Data.”; with *Key_1* and *Key_2* being component symmetric keys for that purpose. *Key_1* and *Key_2* **shall** be generated and/or established independently according to the rules for component symmetric keys from NIST SP 800-133 Rev. 2, Sec. 6.3. The module **shall** check explicitly that *Key_1* ≠ *Key_2*, regardless of how *Key_1* and *Key_2* are obtained. See section **Additional Comments** below for further implementation considerations. In addition, the CST testing lab **shall** document in TE02.20.01 of the Test Report how the module meets the above requirement.

Additional Comments

1. This interpretation of the IEEE standard is consistent with the requirements on the generation of secret keys for other NIST approved cryptographic algorithms, namely, from a cryptographically strong pseudorandom source or approved KDF and with support from a good entropy source.
2. The check for *Key_1* ≠ *Key_2* **shall** be done at any place BEFORE using the keys in the XTS-AES algorithm to process data with them. This allows for choosing an appropriate place for implementing the check, anywhere from within the algorithm boundary to the module boundary.

C.J Requirements for Testing to SP 800-38G

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

SP 800-38G was published March 2016 and was included in the publication of SP 800-140C in March 2020. SP 800-38G contains the description of two methods, FF1 and FF3, for format-preserving encryption (FPE).

Since the release of this publication, researchers have identified vulnerabilities when the number of possible inputs, i.e., the domain size, is sufficiently small. In response to the [analysis of Durak and Vaudenay on FF3](#), NIST [announced](#) in April of 2017 the intention to revise the FF3 specification by reducing the size of its tweak parameter from 64 bits to 56 bits (in collaboration with the researchers) or to withdraw FF3. It was decided to update FF3 (called FF3-1) to address the vulnerability in a new version of SP 800-38Grev1 (which is still in draft at the time this IG was last revised). This announcement also noted that FF3 is no longer suitable as a general-purpose FPE method.

Question/Problem

Considering the information presented in the Background section of this IG, what are the requirements for claiming compliance to the original version of **SP 800-38G**? Are any portions of **SP 800-38G** available for CAVP testing?

Resolution

CAVP testing is available for the FF1 mode but not the FF3 mode. Therefore, if FF1 is supported by a module in an approved mode of operation, it **shall** be CAVP tested. Vendors **shall** not claim any compliance to FF3 in an approved mode of operation.

In addition, the vendor **shall** document, in the module's Security Policy, the lengths of the following parameters from **SP 800-38G**: *radix*, $radix^{minlen}$, *minlen*, *maxlen*, and *maxTlen*. While **SP 800-38G** requires that the value of $radix^{minlen}$ must be greater than or equal to 100, it is *strongly recommended* that this value be at least one million in order to mitigate guessing attacks and analytic attacks (this aligns with the requirements in the draft **SP 800-38Grev1**).

Additional Comments

1. No special acronym is required in the validation certificate to annotate the module's compliance with **SP 800-38G**. Use "AES" as with any other approved mode of AES.
2. This IG applies to the original version of **SP 800-38G** (March 2016). Once **SP 800-38Grev1** is published, CAVP will have testing available for both the FF1 and FF3-1 modes. Therefore, this IG will be withdrawn following the publication of **SP 800-38Grev1**.

Annex D – Approved sensitive security parameter generation and establishment methods

D.A Acceptable SSP Establishment Protocols

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.09</i>
Relevant Test Requirements:	<i>TE09.09.01-02</i>
Relevant Vendor Requirements:	<i>VE09.09.01-02</i>

Background

Cryptographic modules may use various methods for sensitive security parameter (SSP) establishment within a cryptographic module. These methods include the use of symmetric and asymmetric SSP establishment schemes within protocols to establish and maintain secure communication links between modules. **SP 800-140D** provides a list of approved SSP establishment techniques for establishing keying material that are applicable to FIPS 140-3.

Question/Problem

What are all the types of SSP establishment within a cryptographic module, and what are the approved and allowed methods for each type that may be used in the approved mode of operation?

Resolution

SSP establishment is the process by which critical security parameters (CSP) or public security parameters (PSP) are securely shared either within the module or between two or more entities. This IG lists all types of methods for SSP establishment that may be performed in an approved mode of operation. The specifics of each type of SSP establishment are addressed in the corresponding IGs that this IG references. Therefore, this IG serves as an umbrella IG for the approved and allowed SSP establishment methods.

The following are the six types of methods that may be used in the approved mode for SSP establishment within a cryptographic module.

Key agreement is a method of automated SSP establishment where the resulting keying material is a function of information contributed by two or more participants, so that no party can predetermine the value of the secret keying material independently from the contribution of any other party. Key agreement is performed using key agreement schemes. This procedure is further discussed in [IG D.F](#).

Key transport is a method of automated SSP establishment whereby one party (the sender) selects a value for the secret keying material and then securely distributes that value to another party (the receiver). Key transport is performed using key transport schemes. The approved schemes for key transport that may be implemented within a cryptographic module are referenced in **SP 800-140D** and further discussed in [IG D.G](#), which also lists the allowed key transport schemes.

SSP generation is the process for generating cryptographic SSPs within a cryptographic module. The approved methods for SSP generation are listed in **SP 800-140D**.

SSP entry is a method for SSP establishment where the SSP is entered manually into the module either directly (e.g. keyboard) or electronically (e.g. smart card or local wireless). It does not include the key transport schemes described earlier in this IG. [IG 9.5.A](#) provides further information about mapping SSP entry and output states to the FIPS 140-3 requirements.

Key derivation is a method for deriving keys from the certain parameters using the approved key derivation functions. One possibility is to derive a key from an already existing related key as described in **SP 800-108**. Another is to derive a key for storage applications only, in compliance with **SP 800-132**.

Pre-loading of a key is a method by which a manufacturer of the module can establish a key within the module. A key pre-loaded by the manufacturer is available when the module is first powered-on.

Additional Comments

1. The term sensitive security parameter is defined in **ISO/IEC 19790** as the set of critical security parameters (CSP) and public security parameters (PSP). Examples of CSPs include secret (and private) cryptographic keys as well as authentication data (e.g. passwords, PINs, etc.); examples of PSPs include public cryptographic keys, public key certificates, self-signed certificates.
2. This IG does not address SSP establishment for use in authentication techniques.
3. The SSP establishment method(s) that involve key agreement or key transport used by the cryptographic module **shall** be listed under **AS09.09**.
4. While some IGs referenced from this IG list various Key Agreement and Key Transport methods as either approved or allowed, it is important to keep in mind that the strength of these methods may be weaker than the strength of the transported or agreed-upon key. In this case, the resulting strength of the key should be properly documented. See [IG D.B](#) for ways to calculate the strength of the established key, and [Management Manual - Annex A](#) for the proper way to caveat the possible loss of the established key's cryptographic strength in the module's certificate.

D.B Strength of SSP Establishment Methods

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.10, AS09.11</i>
Relevant Test Requirements:	<i>TE's associated with AS's above</i>
Relevant Vendor Requirements:	<i>VE's associated with AS's above</i>

Background

ISO/IEC 19790:2012 Section 7.9.4 specifies that SSP establishment may consist of

- automated SSP transport or SSP agreement methods or
- manual SSP entry or output via direct or electronic methods.

Automated SSP establishment **shall** [**AS09.10**] use an approved method listed in [SP 800-140D]. Manual SSP establishment **shall** [**AS09.11**] meet the requirements of [Section] 7.9.5.

The SSPs discussed herein include both CSPs and PSP. See [IG D.A](#) for more information related to approved SSP establishment methods.

SP 800-57, [Recommendation for Key Management: Part 1 – General \(Revision 5\)](#), Section 5, Sub-Section 5.6.1.1, Security Strengths of Symmetric Block Cipher and Asymmetric-Key Algorithms, contains Table 1, which provides comparable security strengths for the approved algorithms.

Table 1: Comparable Security Strengths				
Security Strength	Symmetric key algorithms	FFC (e.g. DSA, DH, MQV)	IFC (e.g. RSA)	ECC

				(e.g. ECDSA, EdDSA, DH, MQV)
112	3TDEA	$L = 2048$ $N = 224$	$k = 2048$	$f = 224-255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256-383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384-511$
256	AES-256	$L = 15,360$ $N = 512$	$k = 15,360$	$f = 512+$
<i>Note. Reproduced from SP 800-57, Recommendation for Key Management: Part 1 – General (Revision 5), Section 5, Sub-Section 5.6.1.1, Table 2.</i>				

1. Column 1 indicates the number of bits of security provided by the algorithms and key sizes in a particular row. Note that the bits of security are not necessarily the same as the key sizes for the algorithms in the other columns, due to attacks on those algorithms that provide computational advantages.
2. Column 2 identifies the symmetric key algorithms that provide the indicated level of security (at a minimum), where 3TDEA is specified in **SP 800-67**, and AES is specified in **FIPS 197**. The use of 3TDEA encryption is deprecated through 2023, after which it will be disallowed.
3. Column 3 indicates the minimum size of the parameters associated with the standards that use finite field cryptography (FFC). Examples of such algorithms include DSA as defined in **FIPS 186-4** for digital signatures, and Diffie-Hellman (DH) and MQV key agreement as defined in **SP 800-56A**, where L is the size of the public key, and N is the size of the private key.
4. Column 4 indicates the value for k (the size of the modulus n) for algorithms based on integer factorization cryptography (IFC). The predominant algorithm of this type is the RSA algorithm. RSA is specified in ANSI X9.31 and the PKCS#1 document. These specifications are referenced in **FIPS 186-4** for digital signatures and **SP 800-56B** for SSP establishment. The value of k is commonly considered to be the key size.
5. Column 5 indicates the range of f (the size of n, where n is the order of the base point G) for algorithms based on elliptic curve cryptography (ECC) that are specified for digital signatures in ANSI X9.62 and adopted in **FIPS 186-4**, and for SSP establishment as specified in ANSI X9.63 and **SP 800-56A**. Edwards-curve Digital Signature Algorithm (EdDSA) is specified in **FIPS 186-5**. The value of f is commonly considered to be the key size.

For example, if a 256-bit AES secret key is to be transported utilizing RSA, then $k=15360$ for the RSA key pair. A 256-bit AES key transport key could be used to wrap a 256-bit AES key.

For key strengths not listed in Table 1 above, the correspondence between the length of an RSA or a Diffie-Hellman key and the length of a symmetric key of an identical strength can be computed as:

If the length of an RSA key L (this is the value of k in the fourth column of Table 1 above), then the length x of a symmetric key of approximately the same strength can be computed as:

$$x = \frac{1.923 \times \sqrt[3]{L \times \ln(2)} \times \sqrt[3]{[\ln(L \times \ln(2))]^2 - 4.69}}{\ln(2)} \quad (1)$$

If the lengths of the Diffie-Hellman public and private keys are L and N, correspondingly, then the length y of a symmetric key of approximately the same strength can be computed as:

$$y = \min(x, N/2), \quad (2)$$

where x is computed as in formula (1) above.

Question/Problem

In the context of **SP 800-57**, what should be done to mitigate the risk of compromising the security of SSP establishment methods?

Resolution

The requirement applies to the SSP establishment methods found in **ISO/IEC 19790:2012** section 7.9.4.

If an SSP is established via an SSP agreement or SSP transport method, the transport SSP or SSP agreement method **shall** be of equal or greater strength than the SSP being transported or established. For example, it is acceptable to have a 2048-bit RSA key (112-bit strength) transported using an AES key.

If the comparable strength of the largest SSP (taken at face value) that can be established by a cryptographic module is greater than the largest comparable strength of the implemented SSP establishment method, then the module certificate and Security Policy will be annotated with, in addition to the other required caveats, the caveat "(SSP establishment methodology provides xx bits of encryption strength)" for that SSP establishment method. For example, a 256-bit AES secret key is to be transported utilizing **SP 800-56B rev2**'s RSA key transport, with a value of $k=2048$ for the RSA key pair, the caveat would state KTS-RSA (key wrapping, SSP establishment methodology provides 112 bits of encryption strength).

Furthermore, if the module supports, for a particular SSP establishment method, several SSP strengths, then the caveat will state either the choice of strengths provided by the SSPs while operated in an approved mode, if there are only two possible effective strengths, or a range of strengths if there are more than two possible strengths.

For example, if a module implements 2048 and 3072-bit public key Diffie-Hellman KAS compliant to **SP 800-56A rev3**, with the private keys of 224 and 256 bits, then the caveat would state "KAS (Cert. #A.72; SSP establishment methodology provides 112 or 128 bits of encryption strength)".

In addition, if a module implements, in support of a non-approved but allowed key wrapping protocol, the RSA encryption/decryption with the RSA keys of 2048, 4096 and 15360 bits, and this scheme can be used to transport keys greater than 112 bits in strength (e.g. 128-bit AES), then the caveat would say "RSA (key wrapping; SSP establishment methodology provides between 112 and 256 bits of encryption strength)". These caveats provide clarification to Federal users on the actual strength the module is providing even though Table 2 below states that the strength is sufficient.

Additional Comments

As of the publishing of this document, **SP 800-57 Part 1**, is not addressed in **SP 800-131A Rev 2**, but a future revision is expected to address considerations towards transitioning the minimum security strength up to 128 bits in 2030. The proceeding guidance may be used as forward planning.

SP 800-57, [*Recommendation for Key Management: Part 1 – General \(Revision 5\)*](#) (May 2020) also provides the following information in Section 5.6.3:

Table 2 provides a projected time frame for applying cryptographic protection at a minimum security strength. Between 2011 and 2030, a minimum of 112 bits of security **shall** be provided. Thereafter, at least 128 bits of security **shall** be provided.

1. Column 1 is divided into two sub-columns. The first sub-column indicates the security strength to be provided; the second sub-column indicates whether cryptographic protection is being applied to data (e.g., encrypted) or whether cryptographically protected data is being processed (e.g., decrypted).
2. Columns 2 and 3 indicate the time frames during which the security strength is either acceptable, OK for legacy use, or disallowed.
 - "Acceptable" indicates that the algorithm or key length is currently considered to be secure.
 - "Legacy use" means that an algorithm or key length may be used because of its use in legacy applications (i.e., the algorithm or key length can be used to process cryptographically protected data).

- “Disallowed” means that an algorithm or key length **shall** not be used for applying cryptographic protection (e.g., encrypting) and cannot be used in an approved service.

Table 2: Security strength time frames			
Security Strength		Through 2030	2031 and Beyond
< 112	Applying protection	Disallowed	
	Processing	Legacy Use	
112	Applying protection	Acceptable	Disallowed
	Processing		Legacy Use
128	Applying protection and processing information that is already protected	Acceptable	Acceptable
192		Acceptable	Acceptable
256		Acceptable	Acceptable
Note Reproduced from SP 800-57, Recommendation for Key Management: Part 1 – General (Revision 5), Section 5, Sub-Section 5.6.3, Table 4.			

The algorithms and key sizes in the table are considered appropriate for the protection of data during the given time periods. Algorithms or key sizes not indicated for a given range of years **shall** not be used to protect information during that time period. If the security life of information extends beyond one time period specified in the table into the next time period (the later time period), the algorithms and key sizes specified for the later time **shall** be used.

D.C References to the Support of Industry Protocols

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	
Relevant Test Requirements:	
Relevant Vendor Requirements:	

Background

The cryptographic modules may implement various protocols known in the security industry. The examples of such protocols are IKE, TLS, SSH, SRTP, SNMP and TPM, with their KDFs listed in **SP 800-135rev1**. These protocols usually include a complete or partial SSP establishment scheme and, sometimes, an encrypted session that uses the newly established key to protect sensitive data.

The Security Policy may make references to modules’ support of such protocols. This IG provides guidance to how and under what conditions the protocol references should be documented.

Question/Problem

What are the module documentation requirements to show support for the protocols which have their key derivation functions listed in **SP 800-135rev1**?

Resolution

FIPS 140-3 and its SP 800-140 series do not address protocols. Only the cryptographic *algorithms* (such as, for example, AES or ECDSA) and *schemes* (such as the key agreement schemes from **SP 800-56A** or the RSA-based key encapsulation schemes from **SP 800-56B**) that are approved and allowed may be used in the approved mode of operations. These algorithms and schemes are referenced in **SP 800-140C** and **SP 800-140D**.

The protocols' KDFs described in **SP 800-135rev1** are well-defined and are viewed as algorithms, not protocols within the scope of a FIPS 140-3 validation. The CAVP testing for such KDFs is available. The testing laboratories **shall** determine if any of the KDFs implemented in the module are the same as those described in **SP 800-135rev1**.

There are four possible implementation and documentation cases as follows:

1. If the module implements a KDF from **SP 800-135rev1** and this KDF has not been validated by the CAVP, then the module's certificate **shall not** list this function. The module's Security Policy **shall** make it clear that the corresponding protocol **shall not** be used in an approved mode of operation. In particular, none of the keys derived using this key derivation function can be used in the approved mode.
2. If the module implements a KDF from **SP 800-135rev1** and this KDF has been validated by the CAVP, then the module's certificate **shall** list the KDF on the approved algorithm line as a CVL entry. If the module's Security Policy claims that the module supports or uses the corresponding protocol, then the Security Policy **shall** state that no parts of this protocol, other than the approved cryptographic algorithms and the KDFs, have been tested by the CAVP and CMVP.
3. If the module does not implement any KDFs from **SP 800-135rev1** but the module's Security Policy claims that the module supports or uses parts of the corresponding protocol(s) then no entry on the certificate's approved or allowed algorithms lines is required. As in the case considered above (2), the Security Policy **shall** state that this protocol has not been reviewed or tested by the CAVP and CMVP.

This situation may occur when a module implements a portion of a protocol, e.g. not including the KDF, and it is the calling application's responsibility to perform the entire protocol.

4. If the module does not implement a KDF from **SP 800-135rev1** and the module's Security Policy makes no claims that the module supports or uses any of the protocols named in **SP 800-135rev1** then the rules explained in this IG do not apply. The module may implement a (non-**SP 800-135rev1**) SSP establishment scheme if it meets the applicable requirements of [IG D.F](#) and [IG D.G](#).

Additional Comments

1. The use of KDFs described in **NIST SP 800-108** and **NIST SP 800-56C** are out scope for the purposes of this IG.

D.D Elliptic Curves and the FFC Safe-Prime Groups in Support of Industry Protocols

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.10</i>
Relevant Test Requirements:	<i>TE09.10.01-02</i>
Relevant Vendor Requirements:	<i>VE09.10.01</i>

Background

Various industry protocols employ SSP establishment techniques. These techniques commonly use the Diffie-Hellman schemes, utilizing either the Finite Field (FFC) or the Elliptic Curve (ECC) cryptography. Over the years, the protocols developed a notation of their own to define the sets of the Diffie-Hellman domain parameters and the specific elliptic curves. This notation is often different from the corresponding terminology used **FIPS 186-4**, **SP 800-56A** and in the FIPS 140-3 Implementation Guidance. This results in difficulties when establishing a module's compliance with the CMVP validation requirements and in understanding the SSP establishment scheme's encryption strength as expressed in the terminology adopted for the FIPS 140-3 Implementation Guidance.

It is therefore necessary to establish an unambiguous correspondence between the Finite Field Diffie-Hellman domain parameters as defined in **SP 800-56A** and those documented as the specific Modular Exponential (MODP) or Finite Field Diffie-Hellman Ephemeral (FFDHE) FFC groups used in various publications such as the IETF RFCs. Similarly, a mapping has to exist between the NIST Recommended Curves defined in **FIPS 186-4** (and referenced in **SP 800-56A**) and those commonly used in various industry protocols.

Question/Problem

1. What is the relationship of the Diffie-Hellman domain parameters used in **SP 800-56A** and those defined by the FFC safe-prime groups (MODP and FFDHE)?
2. To which NIST recommended curves do the curves used in various industry protocols correspond?

Resolution

The FFC Diffie-Hellman safe-prime groups used in industry protocols such as the Internet Key Exchange protocol (IKE) or the Transport Layer Security protocol (TLS) employ the so-called “safe” primes; that is, $p = 2q + 1$. Per Appendix D of **SP 800-56A**, only the safe primes defined in Sections 3-7 of RFC 3526 (IKE) and in Appendix A of RFC 7919 (TLS) may be used in these protocols. The former prime groups are named MODP in RFC 3526 and are shown in Table 25 in **SP 800-56A**. The latter prime groups are named FFDHE in RFC 7919 and are shown in Table 26 in **SP 800-56A**.

The SSHv2 protocol, as defined in RFC 4253, employs the “diffie-hellman-group14-sha1” key exchange method, which is based on the use of the 2048-bit MODP group.

The elliptic curves used in certain industry standards and the corresponding NIST Recommended Curves are listed in Table 24 of **SP 800-56A**.

All curves listed in Table 24 may be used either in the approved key agreement schemes (if all other applicable requirements are met) or in the allowed schemes. See [IG D.F](#) for more information.

Curve or group additions to the tables shown in Appendix D of **SP 800-56A** may be added to this Implementation Guidance as applicable.

Additional Comments

1. For the purposes of this Implementation Guidance, an industry protocol is either defined or documented and supported by one of the well-recognized standards bodies, such as the IEEE, IETF, ANSI or ISO SC27 (the list is not exclusive). The MODP or FFDHE groups and the various elliptic curves referenced in this Implementation Guidance as protocol-specific have been defined in the IETF RFC publications.
2. This Implementation Guidance does not discuss the Oakley Groups. The reader may refer to the IETF RFC 2409 for more information.
3. While **FIPS 186-4** is a digital signature standard, some of its provisions also apply to the SSP establishment standards, such as **SP 800-56A**.
4. The **SP 800-56A** notation in this Implementation Guidance refers to the latest publication of the standard: NIST Special Publication [800-56A Rev3](#).

D.E Assurance of the Validity of a Public Key for SSP establishment

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.10</i>
Relevant Test Requirements:	<i>TE09.10.01-02</i>
Relevant Vendor Requirements:	<i>VE09.10.01</i>

Background

The correct functioning of public key algorithms depends, in part, on the arithmetic validity of the public key.

Both the owner and the recipient of a public key need to obtain assurance of public key validity before using the key for operational purposes after SSP establishment. Public key algorithms for SSP establishment are specified in **SP 800-56A** and **SP 800-56B**. Methods for obtaining assurance of public key validity are provided in Section 5.6.2 of **SP 800-56A**, and in Section 6.4 of **SP 800-56B**.

The SSP establishment schemes in **SP 800-56A** are specified using either static (long term, multi-use) keys or ephemeral (short term, single use) keys or both. The keys used in the **SP 800-56B** schemes are generally long term (i.e., static) keys.

Since a static key is normally used for a relatively long period of time, and a number of methods are provided for obtaining assurance of public key validity either by the owner or recipient directly, or by using a trusted third party, the process of obtaining the assurance is not too onerous. However, methods for obtaining this assurance for ephemeral keys are more limited, since a trusted third party is normally not available for obtaining the required assurance. The owner of an ephemeral public key generates that key, and obtains assurance of ephemeral public key validity by virtue of generating the key as specified in **SP 800-56A** (see Section 5.6.2.1; Note that this section applies to the owner assurances of both Static and Ephemeral public key validity). However, the recipient of an ephemeral public key must obtain the assurance by performing an explicit public key validation process.

Question/Problem

Public key validation requires a certain amount of time to perform, which can significantly affect communication performance. Can this process be omitted if at least some of the security goals (i.e., authentication of the public key owner and the integrity of the ephemeral key) are fulfilled by other means?

Resolution

The owner or a recipient of a static public key **shall** obtain assurance of the validity of that public key using one or more of the methods specified in **SP 800-56A** or **SP 800-56B**, as appropriate. The owner of an ephemeral public key **shall** obtain assurance of the validity of that key as specified in **SP 800-56A**. Explicit public key validation of an ephemeral public key is required as specified in **SP 800-56A** by a recipient, except in the following situation; in this case, explicit public key validation of the ephemeral public key by the recipient is optional:

1. The ephemeral public key was generated for use in an FFC dhEphem key agreement scheme or an ECC Ephemeral Unified Model key agreement scheme, and
2. The key agreement scheme is being conducted using a protocol that authenticates the source and the integrity of each received ephemeral public key by means of an approved security technique (e.g., a digital signature or an HMAC).

Protocols that satisfy #2 above and, therefore, may omit the explicit ephemeral public key validation process include:

- Internet Key Exchange protocol, versions IKEv1 and IKEv2,

- Transport Layer Security (TLS) protocol and Datagram Transport Layer Security (DTLS) protocol, versions 1.0 and higher.

Additional Comments

1. In this Guidance, both **SP 800-56A** and **SP 800-56B** refer to all published versions of the corresponding standards, unless explicitly stated otherwise.
2. If a cryptographic module implements a key agreement / shared secret computation scheme whereby the recipient of an ephemeral public key omits the explicit ephemeral public key validation, the modules Security Policy **shall** indicate the appropriate protocol listed above that allows the omission of the validation in order to claim conformance to this Implementation Guidance.
3. **SP 800-56A Rev3** provides a choice of how key validation may be performed when it is required by the standards. The module may perform either the full validation or, if applicable, a partial validation of an ephemeral public key. The vendor may consider performing the partial ephemeral public key validation even if in those cases when this Implementation Guidance provides an exemption from the public key validation requirement.
4. For the FFC schemes, a partial public key validation applies only when using “safe” primes. This includes all schemes claiming compliance with the most common IETF protocols.

D.F Key Agreement Methods

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Transition End Dates	
Relevant Assertions:	<i>AS09.10</i>
Relevant Test Requirements:	<i>TE09.10.01-02</i>
Relevant Vendor Requirements:	<i>VE09.10.01</i>

Background

Cryptographic modules may implement various sensitive security parameter (SSP) establishment schemes to establish and maintain secure communication links between modules. SSP establishment includes the processes by which secret keying material is securely established between two or more entities. Keying material is data that is necessary to establish and maintain a cryptographic keying relationship. These schemes are classified into key agreement schemes and key transport schemes. Key transport is addressed in [IG D.G](#); this IG addresses key agreement.

Key agreement is a method of SSP establishment where the resultant key is a function of information contributed by two or more participants, so that no party can predetermine the value of the key independently of the other party's contribution using automated methods. Key agreement is performed using key agreement schemes.

Question/Problem

What are the approved and allowed key agreement techniques that can be used in an approved mode of operation?

Resolution

There are various *scenarios* for the full or partial **key agreement schemes** that have been approved and/or allowed for use in the approved mode of operation.

Approved methods for key agreement.

Scenario 1 Approved **SP 800-56B Rev2**-compliant RSA-based key agreement schemes. The module's implementation of a key agreement scheme **shall** show compliance with **SP 800-56B Rev2**.

The implementations claiming compliance to **SP 800-56B Rev2** may choose one of the following two paths, or both:

(1) A CAVP-tested compliance with the derivation of a shared secret Z in one of the schemes in Sections 8.2 and 8.3 of **SP 800-56B Rev2**. This compliance will be annotated as **KAS-RSA-SSC**. If compliance for party U with the KAS1-basic scheme from Section 8.2.2 is claimed, then the generation of C will be tested. In all other cases, the CAVP test will verify the correctness of the value of Z .

(2) A tested compliance with one or both RSA-based schemes KAS1 and KAS2, defined, respectively, in Sections 8.2 and 8.3 of **SP 800-56B Rev2**. The implemented schemes may be either "basic", shown in Sections 8.2.2 and 8.3.2, or include the key confirmation per sections 8.2.3 and 8.3.3 of **SP 800-56B Rev2**.

When path (2) is chosen, the CAVP testing may be performed either end-to-end, in which case the vendor is issued a **KAS-RSA** algorithm certificate¹¹, or split into (i) testing the computation of the shared secret, (ii) testing the key derivation function used in deriving the keying material, and, if applicable, (iii) testing the key confirmation step in Figure 7, 9, 10, or 11 in **SP 800-56B Rev2**. The key derivation function **shall** comply to either **SP 800-56C Rev1 or Rev2**, in which case this compliance will be documented as an algorithm, annotated as **KDA** in the module's certificate, or to one of the key derivation functions included in **IG 2.4.B**, in which case the compliance will be shown as a **CVL**. Testing of the key confirmation functionality will be shown as a **CVL**. The module's Security Policy **shall** state which key agreement algorithms and algorithm components have been implemented and CAVP-tested.

The shared secret computation portion of a key agreement scheme includes, as applicable, the generation of the domain parameters, key pair generation, computing the RSA primitive(s) (based either on the RSA exponentiation or the use of the Chinese Remainder Theorem (CRT)) used by the module, and performing the public key validation either by the owner of the key pair or by the recipient of the public key.

The Cryptographic Algorithm Self-Test (CAST) for a solution complying with path (1) above **shall** consist of either (when the module takes the role of party U in the KAS1-basic scheme) verifying the computation of an RSA primitive referenced in Action 1 in Section 8.2.2 of **SP 800-56B Rev2**, or (except for party U in the KAS1-basic scheme) performing and verifying the correctness of the computation of the shared secret Z . It is sufficient to perform one CAST for each implementation of an **SP 800-56B Rev2**-compliant solution, even if the implementation includes multiple shared secret computation and key agreement schemes.

The RSA parameters in the CAST, such as the modulus length and the private and public exponents, **shall** have the sizes consistent with those supported by the module. If the CAST includes an exponentiation using a random value z submitted to the test, the values m , z and n in the computation of $m^z \pmod n$, in the **SP 800-56B Rev2** notation, **shall** be such that $m^z > n$.

The CAST for a solution complying with path (2) **shall** consist of either performing the CAST(s) acceptable for path (1) for the implemented **SP 800-56B Rev2** shared secret computation scheme(s) followed by the CAST of a key derivation function used in the derivation of the keying material, or of verifying the value of the derived keying material for at least one implemented **SP 800-56B Rev2**-compliant key agreement scheme. All key derivation functions not included in the **SP 800-56B Rev2** CASTs **shall** have their own self-tests prior to their first operational use.

Scenario 2. Approved **SP 800-56A Rev3**-compliant Discrete Logarithm Cryptographic (DLC)-based key agreement scheme. The module's implementation of a key agreement scheme **shall** show compliance with **SP 800-56A Rev3**. The implementations claiming compliance to this scenario may choose one of the following two paths, or both:

¹¹ A KAS-RSA certificate will only be issued if a key-agreement scheme implements a key derivation function from **SP 800-56C Rev1 or Rev2**.

(1) A CAVP-tested compliance with the derivation of a shared secret Z in one or more of the key agreement schemes in Section 6 of **SP 800-56A Rev3**. This compliance will be annotated as **KAS-SSC** in the module's validation certificate.

(2) A tested compliance with one or more of the key agreement schemes in Section 6 followed by the derivation of the keying material as shown in Section 5.8 of **SP 800-56A Rev3**. This may optionally be followed by the unilateral or bilateral key confirmation shown in Section 5.9 of **SP 800-56A Rev3**.

When path (2) is chosen, the CAVP testing may be performed either end-to-end, in which case the vendor is issued a **KAS** certificate¹², or it may be split into (i) testing the computation of the shared secret, (ii) testing the key derivation function used in deriving the keying material, and, if applicable, (iii) testing the key confirmation step in Sections 5.9.1 or 5.9.2 of **SP 800-56A Rev3**. The key derivation function **shall** comply to either **SP 800-56C Rev1 or Rev2** or to one of the key derivation functions included in **IG 2.4.B**. In the former case, this compliance will be documented as an algorithm, annotated as **KDA** in the module's certificate. In the latter case, the compliance will be shown as a **CVL**. Testing of the key confirmation functionality will be shown as a **CVL**. The module's Security Policy **shall** state which key agreement algorithms and algorithm components have been implemented and CAVP-tested.

The shared secret computation portion of a key agreement scheme includes, as applicable, the domain parameter generation (if using the **FIPS-186** primes) or selection, public key validation, key pair generation, the computation of the Diffie-Hellman or MQV primitives using the Finite Field or Elliptic Curve arithmetic, the key confirmation and an implementation of some of the schemes listed in Section 6 of **SP 800-56A Rev3**.

The CAST for a solution complying with path (1) above **shall** consist of verifying the correctness of the computation of the shared secret Z in at least two of the schemes listed in Section 6 of **SP 800-56A Rev3**: one CAST for the Finite Field Cryptography (FFC) methods and one CAST for the Elliptic Curve Cryptography methods, if both the FFC and ECC methods are implemented; otherwise, just one. No separate CASTs are required to test Diffie-Hellman and MQV schemes.

If a CAST is performed for one of the FFC Diffie-Hellman schemes, the parameters p , g and x , in the notation of **SP 800-56A Rev3**, **shall** be chosen such that $g^x > p$, in order to check the modular arithmetic operation. The size of p **shall** be among those supported by the module.

For the ECC Diffie-Hellman CAST, it is sufficient to choose any NIST-recommended curve supported by one the module's ECC-based key shared secret computation schemes, select any point Q in the subgroup of size n of points on that curve and verify the correct computation of the x -coordinate of point $P = hdQ$, with $2 \leq d \leq n-2$, and the parameters n and h are defined as in Section 3.2 of **SP 800-56A Rev3**.

If performing a CAST for an MQV scheme, the parameter p (for FFC) and the curve (for ECC) **shall** also be among those supported by the module. The points on the curve selected for this self-test need to be in the correct subgroup. The values d , representing the private keys, used in the Section 5.7.2.3 of the **SP 800-56A Rev3** ECC MQV primitive need to be between 2 and $n-2$.

The CAST for a solution complying with path (2) **shall** consist of either a CAST described above for path (1) for the implemented **SP 800-56A Rev3** shared secret computation schemes followed by the CAST of a key derivation function used in the derivation of the keying material, or of verifying the value of the derived keying material for one or more implemented key agreement schemes (Section 6 together with Section 5.8 of **SP 800-56A Rev3**). At least one FFC-based and one ECC-based shared secret computation scheme **shall** be self-tested, if both the FFC and ECC methods are implemented; otherwise, just one. All key derivation functions not included in the **SP 800-56A Rev3** CASTs **shall** have their own self-test prior to their first operational use.

Allowed methods for key agreement.

¹² A KAS certificate demonstrating compliance with Scenario 2 will only be issued if a key-agreement scheme implements a key derivation function from **SP 800-56C Rev1 or Rev2**.

Scenario 3. An ECC scheme using the elliptic curves compliant with [IG C.A](#). This scheme **shall** be shown as *allowed* in the module's Security Policy and documented on the certificate's non-approved line. It is vendor's responsibility to demonstrate that

- (a) the curve(s) are compliant with [IG C.A](#),
- (b) the rules of **SP 800-56A Rev3** have been followed whenever possible, given that the curves may not be defined in a NIST publication, and
- (c) the module supports Scenario 2 above using at least one NIST-recommended curve.

No additional CASTs are required for Scenario 3, since the use of the key agreement schemes under this scenario is not approved but allowed. Moreover, the use of this scenario implies testing Scenario 2 for at least one NIST-recommended elliptic curve, including the execution of the corresponding self-tests.

Additional Comments

1. This IG does not address SSP establishment techniques other than those used for key agreement.
2. The SSP establishment method(s) used by the cryptographic module **shall** be listed under **AS09.10**.
3. For Scenario 1, KAS1 may be implemented as either a basic scheme (no key confirmation) or a Party_V-Confirmation scheme. KAS2 may be implemented as either a basic, or a Party_V-Confirmation, or a Party_U-Confirmation or a bilateral-confirmation scheme. The module's Security Policy **shall** state which of the following schemes have been implemented and tested.
4. The FIPS 140-3 certificate annotation examples for the key agreement schemes can be found in the [Management Manual - Annex A](#).
5. The module **shall** obtain the appropriate assurances, as required in Sections 5.6.2 of **SP 800-56A Rev3** and 6.4 of **SP 800-56B Rev2**.
6. If a full key agreement scheme is implemented (path (2) in Scenarios 1 and 2) and its components are tested separately by the CAVP as shown by the (i) – (ii) – (optionally) (iii) sequence in these scenarios, then the module's certificate will show the **KAS-RSA** or **KAS** entries and reference the applicable tested components within these entries. The certificate will also list these tested components as individual entries.
7. There is no self-test requirement for the key confirmation functionality.
8. There is no requirement to perform the CASTs for both the RSA exponentiation and the CRT methods. Separate CASTs are required to test the FFC and ECC methods, if the module supports both: the difference between them is far more substantial than that among the various RSA techniques. No separate CASTs are required to test the Diffie-Hellman and the MQV schemes.
9. Every module that implements a full key agreement scheme **shall** use only the approved key derivation functions documented in **SP 800-56C Rev1** or **Rev2** or in [IG 2.4.B](#). Note that all **SP 800-135 Rev1** KDFs and the TLS 1.3 KDF are included in [IG 2.4.B](#).
10. If CAVP testing to **SP 800-56A Rev3** is unavailable at the time this Implementation Guidance is published, vendors may claim vendor affirmation to **SP 800-56A Rev3** per FIPS 140-2 IG D.1-rev3 until testing becomes available.
11. The acronym **SSC** stands for “shared secret computation”.

D.G Key Transport Methods

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>

Last Modified Date:	<i>September 21, 2020</i>
Transition End Dates	<i>12/31/2023 – See Below</i>
Relevant Assertions:	<i>AS09.10</i>
Relevant Test Requirements:	<i>TE09.10.01, TE09.10.02</i>
Relevant Vendor Requirements:	<i>VE09.10.01</i>

Background

Cryptographic modules may implement various sensitive security parameter (SSP) establishment schemes to establish and maintain secure communication links between modules. SSP establishment includes the processes by which secret keying material is securely established between two or more entities. Keying material is data that is necessary to establish and maintain a cryptographic keying relationship¹³. These schemes are classified into key agreement schemes and key transport schemes. Key agreement is addressed in [IG D.F](#); this IG addresses key transport.

Key transport is a method of SSP establishment whereby one party (the sender) selects a value for the secret keying material and then securely distributes that value to another party (the receiver). Key transport can be provided using either symmetric or asymmetric techniques.

Question/Problem

What are the approved and allowed key transport techniques that can be used in an approved mode of operation?

Resolution

Symmetric and asymmetric algorithms are used to provide confidentiality and integrity protection of the keying material to be transported. Key transport includes some means of key encapsulation or key wrapping for the keying material to be transported. Key transport **shall** be performed using the appropriate key lengths classified as acceptable, deprecated or legacy-use as specified in [SP 800-131Arev2](#).

Key Encapsulation is a class of techniques whereby keying material is encrypted using asymmetric (public key) algorithms; integrity protection is also commonly provided. The amount of keying material is usually limited by the practicality of performing the encryption operation. The key used for key encapsulation is called a key encapsulation key, which is a public key for which the associated private key is known by the receiver.

Key Wrapping is a class of techniques whereby keying material is encrypted using symmetric algorithms; integrity protection is also commonly provided. The key used by the key wrapping algorithm to wrap the key to be transported is called a key wrapping key, which is a key that must be known by both the sender and the receiver.

Approved methods for key transport.

- Employing an approved RSA-based key transport scheme, as specified in [SP 800-56Brev2](#). The implemented scheme **shall** be tested to **SP 800-56Brev2**. A KTS-RSA entry **shall** be added to the module's validation certificate as shown in the [Management Manual - Annex A](#). The Security Policy **shall** document the tested RSA modulus sizes, the method (from **FIPS 186-4**) of RSA key generation, the tested key confirmation and assurances, as defined in Sections 5 and 6 of **SP 800-56Brev2**, and whether the encapsulation, un-encapsulation or both methods are supported. If an RSA private key is generated by the module, then the module's validation certificate **shall** include an RSA signature algorithm certificate which would confirm that the RSA prime generation method has been tested. In addition, the Security Policy **shall** indicate the module's support for the KTS-OAEP scheme and, if applicable, document the module's readiness to use the transported key in a hybrid scheme defined in Section 9.3 of **SP 800-56Brev2**.
- Employing a key wrapping key, shared by the sender and receiver, together with an approved symmetric key-wrapping algorithm to wrap the keying material to be transported. Approved key wrapping algorithms are specified in [SP 800-38F](#). One method is to use the AES in either the KW or

¹³ The state existing between two entities when they share at least one cryptographic SSP.

KWP mode, or the Triple-DES in the TKW mode. Another is to use a previously approved authenticated symmetric encryption mode, such as, AES GCM, for key wrapping. Yet another approved key-wrapping technique is a “combination” method: use any approved symmetric encryption mode, such as AES ECB, AES CBC, Triple-DES ECB, etc. together with an approved authentication method (for example, HMAC or AES CMAC, or KMAC). The entire wrapped message **shall** be authenticated. After **December 31, 2023** the use of any mode of the Triple-DES algorithm for key wrapping is disallowed.

The symmetric key encryption algorithm, and, if applicable, the authentication algorithm, used for key wrapping **shall** be tested and validated by the CAVP, and the algorithms’ certificate numbers **shall** be shown on the module’s certificate. If the security strength of the key wrapping algorithm and the wrapping algorithm’s key can be lower than that of the (potential) security strength of the wrapped key, then the resulting security strength of the wrapped key is the security strength of the key wrapping key and algorithm, and **shall** be shown on the module’s certificate in accordance with the [Management Manual - Annex A](#).

Allowed methods for key transport in an approved mode.

- Any RSA-based key encapsulation/un-encapsulation algorithm that only uses a PKCS#1-v1.5 padding scheme and an RSA modulus that is at least 2048 bits long. The PKCS#1-v1.5 padding **shall** be performed as shown in Section 8.1 of RFC 2313. The module’s Security Policy **shall** state that this padding method is used. The testing laboratory **shall** verify this claim by performing a code review and an analysis of an implementation’s logic. This allowance expires on **December 31, 2023**.
- A key *unwrapping* using any approved mode of AES or two-key or three-key Triple-DES. Key wrapping is not allowed if the algorithm does not meet the requirements of **SP 800-38F**.

The use of these algorithms by the module for key transport **shall** be annotated on the certificate’s allowed algorithm line as shown in the [Management Manual - Annex A](#).

The **SP 800-56Brev2** Self-Tests:

When claiming compliance with the RSA algorithms used in the key encapsulation and un-encapsulation schemes described in **SP 800-56Brev2**, the module is required to perform a cryptographic algorithm self-test (CAST). If a known answer test (KAT) is used (rather than a *comparison* test or a *fault-detection* test), and an RSA encryption of keys is supported, the module **shall** have an RSA encryption of a vendor-selected message *M* pre-computed and then, prior to the use of the RSA encryption function, the module **shall** perform the RSA encryption again and compare the newly generated result to the pre-computed value. The bit length of the message **shall** be compatible with the bit length of a string encrypted by the RSA.

If an RSA decryption of keys is supported, the module **shall** have a CAST for the RSA decryption. If a KAT is used, the module **shall** start with a selected value representing a ciphertext and decrypting this value using the RSA algorithm. The result of said decryption operation is compared to a pre-computed result. If an implementation of the RSA decryption supports both a decryption with the private key in the basic format and a decryption with the private key in the CRT (Chinese Remainder Theorem) format, then only one CAST – verifying the correct implementation of either method - is required. These two decryption methods are documented, correspondingly, in Sections 7.1.2.1 and 7.1.2.3 of **SP 800-56Brev2**.

If the module can perform only one of the RSA encryption/decryption operations, say, either the encapsulation or the un-encapsulation of a cryptographic key, then only the self-test that is attributable to this operation is required.

While it may appear that the requirements for the RSA exponentiation encryption and decryption (corresponding to the key encapsulation and key un-encapsulation schemes) CASTs are identical, they are not. The encryption CAST uses the public key exponent *e*, while the decryption CAST uses the private key *d*. The RSA parameters used in a CAST, including the public modulus *N*, and, as applicable, the message *M* or the ciphertext *c*, **shall** have the sizes consistent with those supported by the module. When an exponentiation function is self-tested, the encryption CAST consists of checking the value of $M^e \pmod{N}$, while the

decryption CAST consists of checking the value of $M^d \pmod N$. Therefore, separate CASTs are required to self-test the encryption and the decryption operations, if both are implemented.

If an RSA signature generation algorithm and an approved RSA-based key un-encapsulation scheme are both supported by the module using the same implementation (same hardware, same code for the RSA primitive computations) and the module is performing the signature generation self-test then it is not necessary to also perform a self-test for the key un-encapsulation scheme, as long as the signature generation CAST is performed prior to the first use of the signature generation or key un-encapsulation functions.

Similarly, if the same implementation performs the common functionality for both the RSA signature verification and an approved RSA-based key encapsulation scheme then it is sufficient to perform a CAST just for the signature verification algorithm, as long as the signature verification CAST is performed prior to the first use of the signature verification or key encapsulation functions.

When an RSA key pair is generated, the conditional self-tests of **ISO/IEC 19790:2012** Section 7.10.3.3, as further addressed in [IG 10.3.A](#) **shall** be performed.

Additional Comments

1. This IG does not address SSP establishment mechanisms other than those used for key transport.
 2. The key transport method(s) used by the cryptographic module **shall** be listed under **AS09.10**.
 3. While it may be sufficient, as explained in this Implementation Guidance, to perform only a digital signature self-test and not the key encapsulation/un-encapsulation self-tests, the reverse is not true, and the digital signature algorithm self-tests are always required. The reason is that the self-tests for the RSA-based key transport schemes described in this IG are more limited in scope (they only test the RSA primitives) than the digital signature self-tests.
 4. The module's compliance with either the symmetric or the asymmetric key based approved key transport techniques **shall** be annotated on the validation certificate's Approved Cryptographic Algorithms line as KTS or KTS-RSA, respectively, and in the approved cryptographic algorithms list in the Security Policy, with the caveats, as necessary and as shown in the [Management Manual - Annex A](#).
 5. The use of the allowed methods for key transport **shall** be annotated on the certificate's Allowed Algorithm line as shown in the [Management Manual - Annex A](#), and in the allowed algorithms list in the Security Policy.
 6. As the **SP 800-38F** compliant schemes are comprised of approved algorithms that must be tested and issued validation certificates from the CAVP, no vendor affirmation of this key transport scheme in the module's validation certificate is permitted.
 7. For the **SP 800-38F** schemes, it is the tester's responsibility to verify that when using the "combination" method described above, the entire message gets authenticated.
 8. The key wrapping used in many industry protocols, such as TLS and SSH, is likely to be compliant with one of the provisions of **SP 800-38F** if the keys are provided as part of the protocol payload. If a module implements such a protocol and intends to import or export the keys to or from the module's boundary, then the module **shall** claim key transport in the context of the protocol and document it as a KTS.
 9. This IG closely follows the transition dates as specified in **SP 800-131Arev2**, published in March 2019. However, a difference worth noting is that **SP 800-131Arev2** allows non-**SP 800-56Brev2** compliant key transport methods through 2020. However, FIPS 140-3 does not adopt this transition and all non-**SP 800-56Brev2** key transport methods are disallowed except for the PKCS#1-v1.5 padding as explained in this IG. This is because FIPS 140-3 testing will only be available for a short time period (a few months) before the transition goes into effect by the end of 2020, so there is little value in permitting submissions that will become disallowed come January 2021.
-

D.H Requirements for Vendor Affirmation to SP 800-133

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20</i>
Relevant Test Requirements:	<i>TE02.20.01-2</i>
Relevant Vendor Requirements:	<i>VE02.20.01</i>

Background

Key generation is the process of generating cryptographic keys within a cryptographic module from various input sources. These sources **shall** include an output from an approved random bit generator located within the physical boundary of the cryptographic module that is deriving the key. **SP 800-133** is a recommendation that discusses the generation of the keys to be managed and used by approved cryptographic algorithms. On June 4, 2020, NIST published **SP 800-133 Revision 2** to keep pace with the changing portfolio of standards. **SP 800-133** herein refers to this revision of the standard.

Question/Problem

To claim the *vendor affirmation* to **SP 800-133**, what sections of the publication need to be addressed?

Resolution

To claim the vendor affirmation to **SP 800-133** the vendor **shall** generate the module's symmetric keys and seeds used for generating the asymmetric keys using methods described in Section 4 of **SP 800-133**. If a key generating method involves an XOR, as when generating the bit string $B = U \oplus V$ shown in Section 4 of **SP 800-133**, this step and the nature of the parameters U and V **shall** be explained in detail by the vendor.

Note that the four examples in Section 4 of **SP 800-133** are informative, not normative. The vendor may either affirm that the module follows one of these examples, or demonstrate that the module uses a different method to meet the independence requirement for U and V. The requirement that U is an output of an approved DRBG (updated, possibly, using qualified post-processing, as explained below) is normative.

The module may further generate a symmetric key or a seed used in generating the asymmetric keys as shown in Section 6.3 of **SP 800-133**, provided that

- (a) At least one of the component keys K_1, \dots, K_n is generated as shown in Section 4 of **SP 800-133** with an independence requirement of Section 6.3 met, and
- (b) None of the component keys K_1, \dots, K_n are generated from a password.

The module's test report **shall** explain how the keys K_1, \dots, K_n are generated, how the values D_1, \dots, D_m (if used) are obtained, and present the assurances that the applicable Section 6.3 of **SP 800-133** requirements on these parameters are satisfied.

The module may perform a qualified post-processing, explained in [IG D.I](#), to the output U of an approved DRBG before passing this updated value of U to the key generation process.

Vendor affirmation to **SP 800-133** is required for all methods covered by Sections 4 and 6.3 of this standard; that is, when a symmetric key or a seed for asymmetric key generation is generated starting with a random bit string. The module's validation certificate **shall** have a CKG entry only if the module is generating keys for the symmetric-key algorithms. Only one CKG entry is required for the module's certificate, even if the module employs multiple key generation methods that must be documented in the certificate. The Security Policy **shall** provide the details of each method.

A module's compliance with the key generation methods shown in sections of **SP 800-133** (other than 4 and 6.3) are covered by other standards. If the vendor wishes to claim compliance with sections other than 4 and 6.3 and the CKG entry in the module's certificate is not needed, according to this IG, then the Security Policy

(only; not the validation certificate) **shall** claim the vendor affirmation to **SP 800-133** and provide the details for the reader to understand this claim.

Additional Comments

1. For more information on sensitive security parameter (SSP) establishment methods, see additional [IG D.A.](#)
2. If the module directly uses an output U from an approved DRBG or an output from a post-processing algorithm shown in [IG D.I](#) as a symmetric key or as a seed to be used in the asymmetric key generation, then it is not necessary to explain that this technique is equivalent to XORing of U and V where V is a string of binary zeros. The Security Policy **shall** state how the resulting symmetric key or a seed is generated.
3. Section 6.3 in **SP 800-133 Revision 2** corresponds to Section 6.6 of **SP 800-133 Revision 1**.
4. The method of generating a key by a key-extraction process defined in item 3 of Section 6.3 of **SP 800-133 Revision 2** is new to the **SP 800-133** series. This method is approved for use in the approved mode upon the publication of this Implementation Guidance.

Test Requirements

Code review, vendor documentation review, and mapping of the module's key generation procedures into the methods described in **SP 800-133**.

D.I The Use of Post-Processing in Key Generation Methods

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.06</i>
Relevant Test Requirements:	<i>TE09.06.01-3</i>
Relevant Vendor Requirements:	<i>VE09.06.01-2</i>

Background

FIPS 140-3 Derived Test Requirements (DTR) Section 5.1 states that “approved sensitive parameter generation...methods [are] addressed in SP 800-140D.” **AS09.06** further states that “if an approved...SSP generation...method requires random values, then an approved RBG **shall** be used to provide these values.” For consistency with the current version of **SP 800-133**, this IG refers to cryptographic keys as opposed to SSPs in general.

Question/Problem

The NIST *Recommendation for Cryptographic Key Generation*, **SP 800-133**, does not include the so-called post-processing, which is instead documented in this IG. The remaining key generation methodology is adequately addressed in the latest version of **SP 800-133**.

Why have two separate documents ([IG D.I](#) and **SP 800-133**) to illustrate an almost identical functionality?

Resolution

The vendor has an option to perform a qualified post-processing that would apply to U, an output of an approved DRBG, before the updated value of U is passed to the **SP 800-133**-compliant portion of the key generation process. Post-processing is not shown in **SP 800-133** and, therefore, not addressed in [IG D.H](#).

Qualified Post-Processing

The value of U in the **SP 800-133** key generation mechanism is the output of an approved DRBG. As explained earlier, this DRBG output may be further modified by applying qualified post-processing *before* it is used to compute the secret value B (from Section 4). When post-processing is performed on DRBG output, the output of the post-processing **shall** be used in place of any use of the DRBG output. This output from the post-processing becomes the new U .

Let M be the length of the output requested from the DRBG by a consuming application, and let R_M be the set of all bit strings of length M . When the output is to be used for keys, M is typically a multiple of 64; however, these algorithms are flexible enough to cover any output size. Let R_N be the set of all bit strings of length N , and let $F: R_N \rightarrow \{0, 1, \dots, k-1\}$ be a function on N -bit strings with integer output in the range 1 to k , where k is an arbitrary positive integer. Let $\{P_1, P_2, \dots, P_k\}$ be a set of permutations (one-to-one functions) from R_M back to R_M . The P_j 's may be fixed, or they may be generated using a random seed or secret value. Examples of F and P_i are given below.

Let r_1 be randomly selected from the set R_N (i.e., r_1 is a random N -bit value), and let r_2 be randomly selected from the set R_M (i.e., r_2 is a random M -bit value). Both r_1 and r_2 **shall** be outputs from an approved DRBG, such that $N \leq M$ (the case $r_1 = r_2$ is permissible). The post processor's output is the M -bit string $P_{F(r_1)}(r_2)$.

The apparent complexity of this post-processing should not be of any concern to vendors and testing laboratories. The post-processing step is optional. Vendors are not encouraged to design the post-processing into the cryptographic modules.

Examples of $F(r_1)$ used for Post Processing

The function F may be simple or fairly complex.

Let k be the number of desired permutations, and let r_1 represent an N -bit output of an approved DRBG. Two examples are provided:

1. A very simple example of a suitable F is the following, where k is assumed to be an integer in the range 1 to 2^N .

$$F(r_1) = r_1 \bmod k.$$

Here, r_1 is interpreted as an integer represented by the bit string r_1 .

2. A more complex example is:

$$F(r_1) = \text{HMAC}(\text{key}, r_1) \bmod k,$$

using a hashing algorithm and a fixed key in the HMAC computation. In this case, k could be as large as 2^{outlen} , or as small as 1, where outlen is the length of the hash function output in bits. (Having a single permutation, while permitted, would certainly not require the use of a keyed hash to “choose” it. On the other hand, $k = 2$ might make sense in the right application.)

Note that in both examples, the k permutations are selected with (nearly) equal probability, but this is not a requirement imposed by this post-processing.

Examples of P_i used for Post-Processing.

Depending on the requirements of the application, the P_i may be very simple or quite complex. The security of the key generation method depends on the P_i being *permutations*.

1. An example of a very simple permutation P_i is bitwise XOR with a fixed mask A_i : $P_i(r_2) = (r_2 \text{ XOR } A_i)$, where r_2 and A_i are M -bit vectors. Continuing this example, if there are four such masks ($k = 4$), the simple function $F(r_1)$ that maps r_1 into an integer represented by the two rightmost bits of r_1 (say, ‘01’ corresponds to 1, ‘02’ corresponds to 2, ‘03’ corresponds to 3, and ‘00’ corresponds to 4) could be used to

choose among them. Then the post-processor's output $P_{F(r_1)}(r_2)$ would be $r_2 \text{ XOR } A_{F(r_1)}$. Note that in this example, $2 \leq N \leq M$, where N is the length of r_1 , and M is the length of r_2 .

[This should not be confused with the XORing defined in equation (1) above. The equation in (1) is applied after each of the U and V values is calculated, including any qualified post-processing, if applicable.]

2. A more complex example would be the use of a codebook to affect a permutation. For example, $P_i(r_2) = \text{Triple-DES}(key_i, r_2)$ could be used on a DRBG whose outputs were 64-bit strings (Triple-DES is a deprecated algorithm and is only provided here for illustrative purposes). Similarly, $P_i(r_2) = \text{AES}(key_i, r_2)$ could be used to effect permutations on a DRBG with 128-bit outputs.

Suppose that there are ten 256-bit AES keys ($k = 10$). Let $F(r_1) = \text{SHA256}(r_1) \bmod 10$. Then the post-processed output $P_{F(r_1)}(r_2)$ would be $\text{AES}(key_{\text{SHA256}(r_1) \bmod 10}, r_2)$. Note that in this case, $4 \leq N \leq M$, where N is the length of r_1 , and M is the length of r_2 (the minimum length of r_1 is determined by the modulus value 10, which is represented in binary as 4 bits).

A similar example, but one with a *much* larger value for k , (e.g., $k = 2^{128}$), might use $key_i = \text{SHA256}(128\text{-bit representation of } i)$. Let $F(r_1) = \text{SHA256}(r_1)$. The output $P_{F(r_1)}(r_2)$ of the post-processing would be $\text{AES}(\text{SHA256}(r_1), r_2)$. Note that in this case, $N = M = 128$.

3. An example of a permutation somewhere between these extremes of complexity is a byte-permutation 'SBOX_i', which will be applied to each byte of input, with the final output being the concatenation of the individually permuted bytes:

$$P_i(B_1 || B_2 || \dots || B_{M/8}) = \text{SBOX}_i(B_1) || \text{SBOX}_i(B_2) || \dots || \text{SBOX}_i(B_{M/8})$$

For specificity, suppose that $M = 128$; there are just 2 byte permutations to choose from, SBOX₀ and SBOX₁; and F maps 8-bit strings to their parity:

- $F(r_1) = 0$ if r_1 has an even number of 1's,
- $F(r_1) = 1$ if r_1 has an odd number of 1's. Note that in this case, $N = 8$.

The post-processing output $P_{F(r_1)}(r_2)$, on the input pair r_1 and $r_2 = B_1 || B_2 || \dots || B_{16}$ would be $\text{SBOX}_{\text{parity}(r_1)}(B_1) || \text{SBOX}_{\text{parity}(r_1)}(B_2) || \dots || \text{SBOX}_{\text{parity}(r_1)}(B_{16})$. To complete the example, suppose that the two byte permutations are specified as:

- SBOX₀ = the AES SBOX, and
- SBOX₁ = inverse permutation to the same AES SBOX. See **FIPS 197** for more details.

Additional Comments

1. If the vendor chooses to perform the post-processing, the vendor **shall** explain the details of how it works. If possible, the vendor should map their method into one of the examples shown in this Implementation Guidance.
2. Although some security strength may be lost during post-processing, the loss is small enough to be ignored for the purposes of FIPS 140-3 validation.
3. The post-processing may apply whenever the module generates either a symmetric cryptographic key or a seed to be used when generating the asymmetric keys.

Test Requirements

Code review, vendor documentation review, and mapping of the module's post-processing into the methods described in this Implementation Guidance.

D.J Entropy Estimation and Compliance with SP 800-90B

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.08, AS09.09</i>
Relevant Test Requirements:	<i>TE09.08.01, TE09.08.01 TE09.09.01, TE09.09.02</i>
Relevant Vendor Requirements:	<i>VE09.08.01, VE09.08.01 VE09.09.01, VE09.09.02</i>

Background

Section 7.9.3 of **ISO/IEC 19790:2012** states that “Compromising the security of the SSP generation method which uses the output of an approved RBG (e.g., guessing the seed value to initialise the deterministic RBG) shall [09.08] require at least as many operations as determining the value of the generated SSP.” TE09.08.02 further states that “The tester shall verify the accuracy of any rationale provided by the vendor. The burden of proof is on the vendor; if there is any uncertainty or ambiguity, the tester shall require the vendor to produce additional information as needed.”

With the publication in January 2018 of **SP 800-90B**, which uses the min-entropy measurement of entropy, vendors and testers have received a standard against which they can design, build and test their entropy sources. Modules that are compliant to FIPS 140-3 **shall** use only entropy sources which are approved, such as those compliant to **SP 800-90B**.

Question/Problem

When will the **SP 800-90B** compliance become mandatory?

What does the vendor need to do to claim compliance with **SP 800-90B**?

How **shall** the compliance with the entropy-generation requirements be documented in the module’s validation certificate?

When establishing the source’s compliance with **SP 800-90B**, how **shall** a laboratory test it and verify the vendor’s claims?

Resolution

If a cryptographic module falls under one of the scenarios of [IG 9.3.A](#) that require entropy estimation, then the module **shall** be tested for its compliance with **SP 800-90B** and [IG D.K](#). The requirements represented by the “**shall**” statements in **SP 800-90B** apply and must be tested by the lab, with the possible exceptions as stated below in this Implementation Guidance and in [IG D.K](#). These requirements include running statistical tests on the raw entropy data, as explained in **SP 800-90B**. Statistical testing **shall** be performed using a software tool available at https://github.com/usnistgov/SP800-90B_EntropyAssessment. Besides the statistical testing, a CST laboratory is still responsible for performing a heuristic analysis of the entropy source, as this is required in Section 3.2.2, item 3, of **SP 800-90B**.

When claiming compliance with **SP 800-90B** to meet the requirements of **AS09.08** and **AS09.09**, the testing laboratory **shall** provide a **PDF** addendum to the submitted test report. This addendum **shall** include a detailed logical diagram showing all components of an entropy source and the numerical results of various tests required by **SP 800-90B**. The addendum **shall** contain both a rationale for why the final entropy assessment is consistent with both the **SP 800-90B** statistical tests and the required heuristic analysis of the entropy source, and a description of how the entropy source satisfies all of the **SP 800-90B** ‘**shall**’ statements.

When a cryptographic module is validated for its compliance with **SP 800-90B**, the module’s validation certificate **shall** include the following entry on the approved algorithm line: **ENT**.

Any new validation submission of a cryptographic module that obtains its entropy from a previously-validated embedded module **shall** comply with **SP 800-90B**.

Additional Comments

1. In compliance with **SP 800-90B**, vendors **shall** provide access to the raw outputs of the noise source. The vendor may use special methods (or devices, such as an oscilloscope) that require detailed knowledge of the source to collect raw data. The testing laboratory is required to include a section in the Entropy Test Report to present a rationale why the data collections methods will not alter the statistical properties of the noise source or explain how to account for any change in the source's statistical characteristics and its entropy yield.
2. The requirement 2 of Section 3.2. of **SP 800-90B** about the entropy source being stationary does not have to be met, as long as it can be guaranteed that the source is generating the sufficient amount of entropy even when operating at the lowest entropy yield. If the source may deteriorate to the point when the generation of the sufficient amount of entropy (sufficient to support the claims about the strengths of the generated cryptographic keys) can no longer be guaranteed, the module's Security Policy **shall** explain what action is to be taken.
3. The approved algorithms used in the vetted conditioning components **shall** be tested by the CAVP (if testing is available for them). This is a reiteration of a requirement from Section 3.1.5.1.2 of **SP 800-90B**.
It is recommended that these algorithms undergo the self-tests as specified in Section 7.10 of **ISO/IEC 19790:2012**. However, these tests are not mandatory if an algorithm implementation is used solely in a conditioning component of an entropy generation process.
4. A restart test requirement from Section 3.1.4.3 of **SP 800-90B** needs to be addressed. A failure of a restart test does not automatically disqualify the module from being validated. Should this failure occur, the lab **shall** analyze the reason for a failure of the test and explain how the entropy requirement can be met in light of this failure.
5. For the applicability of entropy testing and for the specific validation certificate caveats, see [IG 9.3.A](#).
6. When entropy source testing to **SP 800-90B** is applicable, the module's Security Policy **shall** document the overall amount of generated entropy and the estimated amount of entropy per the source's output bit.
7. The **SP 800-90B** testing tool's version number will be made available to users of the tool. This version number **shall** be included in the lab's Entropy Test Report.
8. The new entry, ENT, on the approved algorithm line does not have an algorithm certificate number. This makes it look different from other entries on the same line. As the process of testing to **SP 800-90B** matures, the CMVP will consider issuing the ENT certificates and maintaining a webpage where the details of the test results for the sources that provide entropy to validated modules will be shown.
9. Should the vendor decide to claim an IID assumption of the samples generated by the noise sources, they will need to provide a rigorous proof in support of this claim. As the majority of the noise sources do not produce the IID events, any IID claim by the vendor will be thoroughly vetted by the validation body. A claim of independence and that of an identical distribution **shall** be substantiated separately. For an independence claim, a deep understanding of the underlying operation of the noise source is required. A claim of an identical distribution of the samples **shall** consider a possible deterioration of the source's entropy generation pattern due to the mechanical or the environmental changes or to the timing variations in human behavior.

Further instructions can be found in Section 3.1.2 of **SP 800-90B**.

D.K Interpretation of SP 800-90B Requirements

Applicable Levels:	<i>All</i>
--------------------	------------

Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS02.20, AS09.08, AS09.09</i>
Relevant Test Requirements:	<i>TE02.20.01, TE09.08.01, TE09.08.02, TE09.09.01, TE09.09.02</i>
Relevant Vendor Requirements:	<i>VE02.20.01, VE09.08.01, VE09.08.02, VE09.09.01, VE09.09.02</i>

Background

SP 800-90B was included in **SP 800-140D** on its initial publication in March 2020. [IG D.J](#) specifies how entropy sources with claims of compliance to **SP 800-90B** will be evaluated and tested within the CMVP program. All newly submitted modules requiring an entropy evaluation must demonstrate compliance to **SP 800-90B**.

Question/Problem

Some ambiguities in the **SP 800-90B** document have been publicly documented. Vendors and testing laboratories would like to know the CMVP's interpretation of the requirements in **SP 800-90B** so that evaluations against these requirements are consistent between laboratories. In addition, vendors producing designs intended to meet these requirements would prefer to experience less risk of eventual non-compliance due to differing interpretations of this document within the FIPS 140-3 validation program.

Resolution

1. For Section 2.2.1, the vendor **shall** justify why all processing occurring within the digitization process does not conceal noise source failures from the health tests or obscure the statistical properties of the underlying raw noise output from this digitization process.

Note: This resolution may impact designs that combine the outputs of multiple copies of the same type of physical noise source (see also Resolution #10). For example, in some designs the XOR of the output of noise source copies may pass most statistical tests, even when the noise source copies are in a failure mode where their outputs are wholly deterministic (and thus entropy free). Designs which include such a digitization step require thorough arguments that the included health tests detect degraded and failure modes of the noise source and that the statistical assessment of the identified raw data is meaningful. One possible approach for such designs is to designate the raw data sample as the outputs of all of the noise source copies present concatenated as a string, and then describe the XOR tree as a first stage of non-vetted conditioning. The tester **shall** provide a detailed description of all digitization processes used within the noise source and describe the format of the raw data that was tested. (See **SP 800-90B** Section 3.2.2 Requirement #3, and Section 4.3 Requirements #1, #6, #7, #8, and #9).

2. For Section 3.1.2, use of the IID-track requires that

“The submitter makes an IID claim on the noise source, based on the submitter’s analysis of the design. The submitter **shall** provide rationale for the IID claim.”

[IG D.J](#) states that

“Should the vendor decide to claim an IID assumption of the samples generated by the noise sources, they will need to provide a rigorous proof in support of this claim. As the majority of the noise sources do not produce the IID events, any IID claim by the vendor will be thoroughly vetted by the validation body. A claim of independence and that of an identical distribution **shall** be substantiated separately. For an independence claim, a deep understanding of the underlying operation of the noise source is required. A claim of an identical distribution of the samples **shall** consider a possible deterioration of the source’s entropy generation pattern due to the mechanical or the environmental changes or to the timing variations in human behavior.”

The testing laboratory **shall** evaluate the technical accuracy and completeness of any IID rationale made by the vendor. If it is not possible for the vendor to produce such a rigorous proof and/or it is not possible for the laboratory to verify the correctness and completeness of the vendor's rationale, then the vendor **shall not** make an IID claim for the noise source. (See **SP 800-90B** Section 3.1.2, Section 3.2.2 Requirement #5, [IG D.J](#)).

3. For Section 3.1.5, all processing of the raw data output from the noise sources that happens before it is ultimately output from the entropy source **shall** occur within a *conditioning chain*: a finite sequence of one or more conditioning components where each conditioning component in the chain receives any input data that is claimed to contain entropy from either the primary noise source (for the first conditioning component in the conditioning chain), or from the previous conditioning component in the conditioning chain (for all other conditioning components). An entropy estimate for the output of each conditioning component making up the conditioning chain **shall** be produced. For each non-vetted conditional component within a chain, an entropy estimate h' , defined in Section 3.1.5.2, **shall** be computed using the statistical tests on the conditioned sequential data set for this component, as specified in Section 3.1.5.2. The entropy source's entropy rate is the entropy rate output from the final conditioning component in the conditioning chain.

Note 1. As stated in Resolution #9 below, if the conditioning function is bijective then the vendor may claim that the entropy of the conditioned output, h_{out} , is equal to the entropy of the input, h_{in} . If claiming this property, it is the responsibility of the vendor and the testing lab to demonstrate that the mapping performed by the conditioning function is indeed bijective. They **shall** describe the set A of random data samples before conditioning¹⁴, the set B of samples after the conditioning, and then show that the mapping of A to B performed by the conditioning function is both injective (the different elements of A map into the different elements of B) and surjective (every element of B has an element of A that maps into it.)

Note 2. In view of the **SP 800-90B**, Section 3.1.5.2, requirements, the output of a non-vetted conditioning component that was not shown by the vendor to be a bijective mapping, cannot be considered "full entropy", so the output of any entropy source whose conditioning chain ends with a non-vetted non-bijective conditioning component cannot be considered "full entropy".

Note 3. If the bijection property can be demonstrated for a non-vetted conditioning component, then the statistical testing *for this conditioning component* described in Section 3.1.5.2 of **SP 800-90B** does not need to be performed. Note that the vendor may choose not to claim the conditioning component's bijective property (even if they can prove that the component possesses this property), and instead perform an analysis of the conditional component's output entropy as specified in Section 3.1.5.2 of **SP 800-90B**.

Note 4. This Resolution does not preclude the inclusion of input data from additional noise sources (see **SP 800-90B** Section 3.1.6) or of supplemental data (see Resolution #6) into vetted conditioning components, as such data is not credited as containing entropy in **SP 800-90B**.

4. For Section 3.1.5, for each conditioning component within the conditioning chain, the vendor **shall** specify:
 - a. the parameter n_{in} , a lower bound for the amount of input data obtained from the primary noise source (for the first conditioning component) or the prior conditioning component in the chain (for all other conditioning components), and
 - b. the parameter h_{in} , a lower bound for the assessed entropy supplied within this data.

Note 1. While the above definition of n_{in} may appear to be different from that in **SP 800-90B**, the definition of n_{in} in this IG reflects the intended meaning of this parameter: the size of the input data string from the primary noise source that is credited with the generation of entropy. Any data input into a conditioning component that is not credited with the generation of entropy does not affect the value of n_{in} . The actual amount of data and entropy provided to each conditioning component per-output may vary, so long as the

¹⁴ If the conditioning component is the first (or only) component in the chain of conditioning components in an entropy source, then set A is the alphabet, as defined in Section 1.3 of **SP 800-90B**.

specified lower bounds are consistently satisfied (See **SP 800-90B** Sections 3.1.5, 3.1.5.1.2, 3.1.6, and Section 3.2.3 Requirements #1 and #4).

Note 2. The output of any conditioning component in a conditioning chain may be used as a source of supplemental information for any vetted conditioning component at the same stage of that conditioning chain or earlier (see also Resolution #6).

5. Conditioning components (described in Section 3.1.5 and its subsections, and in Section 3.2.3) are permitted to retain state between invocations.

Resolution #9 allows the vendor to argue that a conditioning function is bijective, and thus preserves entropy. For non-bijective conditioning functions, this is not in general true. When a non-vetted conditioning component is used **SP 800-90B** Section 3.2.3 Requirement #5 obliges the vendor to justify the appropriateness of any non-vetted conditioning function. In the case that the conditioning component retains state, interactions between this retained state and the input data can additionally reduce the output entropy, as the retained state may allow for interactions between the conditioning component's inputs across different invocations. If a non-vetted conditioning component retains state and the primary noise source is non-independent, then the vendor **shall** provide mathematical evidence that the conditioning component's entropy output is not below its assessed value (h_{out}). (See **SP 800-90B** Section 3.2.3 Requirement #5). This mathematical evidence **may** provide and justify an upper bound for the reduction of the conditioning component's output entropy due to the cancellation of mutual information present in both the data input to the conditioning component and the retained state (for example, if a conditioning component updates its internal state by XORing the prior state with the input data, then inputting the same value into the conditioning component twice, even in different invocations, will cancel the entropy contributed by this value). This mathematical evidence **shall** justify why the reduction of the conditioning component's output entropy due to the cancellation of mutual information present in both the data input to the conditioning component and the retained state does not result in the output entropy of the conditioning component being below its assessed value (h_{out}).

Note 1. A conditioning component is viewed as a fixed conditioning function along with its associated state, for example the CMAC conditioning function coupled with its key.

Note 2. All non-vetted conditioning components (including the bijective ones) are required to meet **SP 800-90B** Section 3.2.3 Requirement #5. None of the "shall" requirements specified in **SP 800-90B** Section 3.2.3 Requirement #5 apply to vetted conditioning components.

6. For Section 3.1.5 and its subsections, a vetted conditioning component may optionally take a finite amount of supplemental data (e.g., data from additional noise sources, a prior output of this conditioning component or any conditioning component later in the conditioning chain, an input counter, a time stamp, etc.) in addition to the data from the primary noise source (for the first conditioning component in the conditioning chain) or from the previous conditioning component in the conditioning chain (for all other conditioning components in the conditioning chain). The presence of supplemental data **shall not** be credited for the purpose of computing h_{in} or n_{in} . (See **SP 800-90B** Sections 3.1.5.1.2 and 3.1.6).
7. For Section 3.1.5.1.1, in order for a conditioning function to qualify as "vetted", it **shall** consist solely of one of the listed vetted functions in this section. A conditioning function that integrates a vetted conditioning function as a subcomponent is not a vetted conditioning function unless the entire conditioning function is equivalent to one of the vetted conditioning functions listed in Section 3.1.5.1.1. A conditioning chain may be made up of a mix of vetted and non-vetted conditioning components (see also Resolutions #3 and #6).
8. For Section 3.1.5 and its subsections, n_w **shall not** be claimed to be greater than n_{in} . The narrowest width for non-vetted conditioning components **shall** be established by analysis of their designs. The tester **shall** describe how application of Appendix E resulted in the reported narrowest internal width values. (See **SP 800-90B** Appendix E).
9. For Section 3.1.5, if the conditioning function can be shown to be bijective, then the vendor may claim that $h_{out} = h_{in}$. If this bijective conditioning function is non-vetted, then its output **shall not** be truncated, as per Section 3.1.5.2. None of the vetted conditioning components are bijective in their anticipated use. Any transform that is reversible is bijective, and the tester **shall** specify a detailed procedure for reversing

any conditioning function that is claimed to be bijective. For example, encrypting the output of the noise source and outputting all of the resulting ciphertext is clearly bijective, as one could decrypt the ciphertext and recover the original data. An example of a non- bijective conditioning function is any function that has a compression ratio greater than 1 for all input data, such as repeated XORing of raw data samples together to produce a single output the same width as the raw data.

10. Section 3.1.6 specifies that multiple copies of the same physical noise source are considered as a single noise source. Combining the outputs of the noise source copies under this provision **shall** be considered part of the digitization process, and so Resolution #1 **shall** apply. (See **SP 800-90B** Section 2.2.1 and Appendix B).
11. For Section 3.1.6, multiple ring oscillators may be treated as “copies”, even in the instance where the design and layout of the ring oscillators vary.
12. The **SP 800-90B** document (including Section 3.2.2) and this IG specify requirements for noise sources. Only the “primary noise source” needs to fulfill the requirements that apply to a “noise source” where the term is unqualified by either “primary” or “additional”.
13. In Section 3.2.2, Requirement #1 states

“The operation of the noise source **shall** be documented; this documentation **shall** include a description of how the noise source works, where the unpredictability comes from, and rationale for why the noise source provides acceptable entropy output.”

Requirement #3 describes how the estimate $H_{submitter}$ must be created:

“Documentation **shall** provide an explicit statement of the expected entropy provided by the noise source outputs and provide a technical argument for why the noise source can support that entropy rate”

The technical argument supporting the expected $H_{submitter}$ value **shall** be based on the vendor’s description of the source of unpredictability within the noise source and how the noise source outputs vary depending on this identified unpredictability. Statistical testing may be used to establish parameters referenced within this argument, but the $H_{submitter}$ value **shall not** be the result of some general statistical testing process that does not account for the design of the noise source.

14. Section 4.3 requires that

“The submitter **shall** provide documentation of any known or suspected noise source failure modes (e.g., the noise source starts producing periodic outputs like 101...01), **and shall include developer-defined continuous tests to detect those failures.**”

If the design integrates the described RCT and APT tests and these tests are shown to not detect the vendor-identified known or suspected noise source failure modes, then the developer **shall** include additional developer-defined continuous testing that does detect the vendor-identified noise source failure modes (irrespective of Section 4.4’s statement that the RCT and APT are the only tests required). The tester **shall** verify that all the vendor-identified known or suspected noise source failure modes are detected by the continuous health tests included within the entropy source. (See **SP 800-90B** Section 4.3, Requirements #1, #7, #8 and #9).

15. For Section 4.3, Requirement #3, when stating the false positive rate (alpha) to satisfy the requirement, the false positive rate may be either the alpha used to generate the cutoffs for the APT/RCT tests or the actual observed false positive rate experienced by this health test when supplied with raw data from the noise source in use. The developer **shall** describe the exact meaning of the specified false positive rate and what the relation is between this false positive rate and any cutoff values used with the health tests.
16. For Section 4.4.2, the cutoff value C for the APT **shall** be no larger than the window size (i.e., $C \leq W$).
17. Many types of noise sources do not produce a constant min entropy per output, but instead produce a min entropy per output that is dependent on some internal state. For such noise sources, $H_{submitter}$ and $H = \min(H_r, H_c, H_l)$ in Sections 3.1.3 and 3.1.4.2 **shall** reflect an entropy bound that can be justified in the average case and/or on a per-symbol basis with high probability. When producing the arguments to meet

the requirements of Section 4.5 for developer-defined health tests, it is acceptable to assume that the noise source produces raw data samples whose per-sample min entropy is equal to the assessed min entropy. (This is consistent with the assumptions used in the analysis of the **SP 800-90B** Adaptive Proportion Test (APT) and the Repetition Count Test (RCT).)

18. For Section 4.5, when using simulation to argue that the developer-provided health test satisfies the requirements of Section 4.5, the developer **shall** specify how the data used within this simulation was created. Possible approaches include using:

- the output data of simulated noise sources experiencing the anticipated failure modes,
- the output data of an actual noise source that is forced into failure modes,
- the output of an actual noise source interleaved with generated data that is statistically similar to the anticipated data output by the noise source in a failure mode, and
- generated data that is expected to be statistically similar to the noise source output combined with generated data consistent with the failure mode being simulated.

To fulfill the Section 4.5 requirements using simulation, at least 1 million rounds of simulation **shall** be used for each simulated health test, and there **shall** be sufficient simulation rounds so that at least five health test failures are observed for each health test. (See **SP 800-90B** Section 4.3, Requirement #1 and Section 4.5).

Additional Comments

1. This Implementing Guidance does not address any issues that may arise when running the statistical tests defined in Sections 5 and 6 of **SP 800-90B** or interpreting their results. The CAVP has published an implementation of these tests¹⁵ that includes many small corrections and enhancements to these tests described in **SP 800-90B**. This site also includes a mechanism for reporting errors in the tool, and to provide proposed fixes.

Some further guidance helping the implementers and the testing laboratories interpret the **SP 800-90B** requirements can be found in [IG D.J.](#)

2. **SP 800-90B** uses the term “submitter” for the party that presents an entropy source to the CMVP for their review of compliance within the scope of the cryptographic module’s validation to FIPS 140-3. To be consistent with previously written Implementation Guidance, this term is substituted here for “vendor”, except in places where the text quotes directly from **SP 800-90B**.
3. The tester **shall** verify that each conditioning component’s implementation is fully consistent with the component’s design. This verification **shall** be performed by means of either running a computerized test developed for testing just the conditioning component (separate from the statistical testing of the noise source) or by the code review. The lab **shall** describe in the Entropy Test Report submitted to the CMVP the chosen method for verifying the correctness of each conditioning component’s implementation.

The requirements in this Additional Comment apply to all conditioning components. While the design of the vetted components and of the non-vetted ones whose bijective properties have been demonstrated may guarantee that a certain amount of entropy will be output, the correctness of the components’ implementations has not been established, thus requiring a separate testing or a code review by the CST labs.

The CAVP testing of the approved cryptographic algorithms used in vetted conditioning components is required per Section 3.1.5.1.2 of **SP 800-90B**. Per [IG D.J.](#), while it is recommended that the module performs the self-tests for these algorithms, this is not mandatory if an algorithm implementation is used solely in a conditional component of an entropy generation process. Note that a vetted conditioning component may include more than an approved cryptographic algorithm. For example, buffering may be performed before a cryptographic algorithm is executed. The requirement for the tester to verify the correctness of a conditioning component’s implementation includes the testing or a

¹⁵ The ACVP implementation of the **SP 800-90B** test tool is available at the URL https://github.com/usnistgov/SP800-90B_EntropyAssessment

code review of the functionality of the component that may not be addressed by the CAVP testing of the approved algorithms.

4. The decision of how the conditioning processing is partitioned into discrete conditioning components in a conditioning chain is established by the vendor. The vendor always retains an option to define the multiple conditioning components as a single function, in which case separate testing of components is not required.

In many circumstances, it may be helpful to identify portions of conditioning that are performed by vetted conditioning functions as discrete conditioning components, as the assessed entropy for non-vetted conditioning components is limited by **SP 800-90B**'s required statistical assessment of the output of non-vetted conditioning components (see **SP 800-90B** Section 3.1.5.2 for details), truncation of the output of non-vetted conditioning components is disallowed (see **SP 800-90B** Section 3.1.5.2), and only vetted conditioning components can integrate input from additional noise sources (see **SP 800-90B** Section 3.1.6) or supplemental data (see Resolution #6).

5. This Implementation Guidance **does not** impose any *technical* requirements not currently stated in **SP 800-90B**. The purpose of this IG is to highlight some of the **SP 800-90B** requirements, show how the highlighted requirements can be satisfied, add certain requirements (such as those in Resolution #1) to avoid the implementation mistakes, and provide some optional flexibility to vendors and testing laboratories.

D.L Critical Security Parameters for the SP 800-90A DRBGs

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.01, AS09.04, AS09.05, AS09.08, AS09.25, AS09.26</i>
Relevant Test Requirements:	<i>TE's associated with AS's above</i>
Relevant Vendor Requirements:	<i>VE's associated with AS's above</i>

Background

The FIPS 140-3 cryptographic module Security Policy **shall** specify all cryptographic keys and CSPs employed by the cryptographic module.

Question/Problem

Which are the critical security parameters that determine the security of the **SP 800-90A** DRBG mechanisms?

Resolution

Section 7.1 of **SP 800-90A** states: “[T]he entropy input and the seed **shall** be kept secret.” Therefore, the entropy input string and the seed **shall** be considered CSPs for all the DRBG mechanisms.

During the instantiation of a DRBG an initial internal state is derived from the seed. The internal state contains administrative information and the working state. As stated in Section 8.3 of **SP 800-90A**, some values of the working state are considered secret values of the internal state. Therefore, they **shall** be considered CSPs as well. These values are listed below:

1. Hash_DRBG mechanism

The values of V and C are the “secret values” of the internal state.

2. HMAC_DRBG mechanism

The values of V and Key are the “secret values” of the internal state.

3. CTR_DRBG mechanism

The values of V and Key are the “secret values” of the internal state.

Additional Requirements

1. The **SP 800-90A** requires that the internal state is protected at least as well as the intended use of the pseudorandom output bits requested by the consuming application. **SP 800-90A** further requires that the DRBG internal state is contained within the DRBG mechanism boundary and **shall not** be accessed by non-DRBG functions or other instantiations of that or other DRBGs.
2. **FIPS 186-5** A.3.3 “Per-Message Secret Number Generation for Deterministic ECDSA” uses the HMAC_DRBG mechanism. In this method, “the private key d is concatenated with the hash of the signed message and used as a seed to instantiate the generation process (i.e., the HMAC_DRBG).” This IG applies to this usage of HMAC_DRBG; specifically, the seed (private key d concatenated with the hash of the signed message), along with V and Key in the internal working state, are considered CSPs. TE09.25.01 **shall** specify how this requirement is met.

D.M Using the SP 800-108 KDFs in an Approved Mode

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.09</i>
Relevant Test Requirements:	<i>TE09.09.01, TE09.09.02</i>
Relevant Vendor Requirements:	<i>VE09.09.01, VE09.09.02</i>

Background

When a key is shared between two entities, it may be necessary to derive additional keying material using the shared key. **SP 800-108** provides Key Derivation Functions (KDFs) for deriving keys from a shared key and not, for example, from a shared secret; in **SP 800-108**, the shared key is called a pre-shared key. The shared key may have been generated, entered or established using any method approved or allowed in an approved mode.

Note that [IG D.A](#) contains SSP establishment methods, and includes KDFs that are used during key agreement to derive keying material from a shared secret, which is the result of applying a Diffie-Hellman or MQV primitive. The keying material may be used as a key directly or to derive further keying material.

Question/Problem

Where do the KDFs from **SP 800-108** fit in the SSP establishment process, and under what conditions can these KDFs be used in an approved mode? Are there any other approved methods for deriving additional keys from a pre-shared key?

Resolution

The role of the **SP 800-108** KDFs is to derive new keys from existing keying material. Therefore, all key derivation methods listed in **SP 800-108** are approved for use in an approved mode if the Key Derivation Key K_I , as introduced in Section 5 of **SP 800-108** has been generated, entered or established using a method approved or allowed for keys in an approved mode. Specific requirements for generating symmetric keys using **SP 800-108** are found in Sec. 6.4 of **SP 800-133 Rev 1**, “Symmetric Keys Derived from a Pre-shared Key.” **SP 800-108** KDFs may not be used to generate asymmetric keys.

Other KDFs that are approved for key derivation from shared keying material are:

1. The KDF specified in the Secure Real-time Transport Protocol (SRTP) defined in Sec. 5.3 of **SP 800-135 Rev 1**. Note that this KDF is only approved when performed in the context of the SRTP protocol.

Additional Comments

1. A key hierarchy as specified in Section 6 of **SP 800-108** may be used.
2. Note that the IEEE 802.11i KDFs are included in **SP 800-108**.

D.N SP 800-132 Password-Based Key Derivation for Storage Applications

Applicable Levels:	<i>All</i>
Original Publishing Date:	<i>September 21, 2020</i>
Effective Date:	<i>September 21, 2020</i>
Last Modified Date:	<i>September 21, 2020</i>
Relevant Assertions:	<i>AS09.09</i>
Relevant Test Requirements:	<i>TE09.09.01 and TE09.09.02</i>
Relevant Vendor Requirements:	<i>VE09.09.01 and VE09.09.02</i>

Background

SP 800-132 was published December 2010 and added to **SP 800-140D** in March 2020. This Special Publication defines the methods and the applicability for password-based key derivation for storage applications.

Question/Problem

To use **SP 800-132** password-based key derivation in an approved mode of operation, what sections of the publication need to be addressed and what are the applicable requirements?

Resolution

CAVP validation of the PBKDF algorithm is required.

In Section 5.4 of that Special Publication, four options (1a, 1b, 2a and 2b) are given for deriving a Data Protection Key from the Master Key. The vendor **shall** specify in the cryptographic module’s Security Policy which option or options are used by the module. The Security Policy **shall** also indicate for each of the following options, if used,

Option 1b – the approved key derivation function (KDF) used;

Option 2a – the approved authenticated encryption algorithm **or** approved authentication technique and approved encryption algorithm used;

Option 2b – the approved authenticated encryption algorithm **or** approved authentication technique and approved encryption algorithm **and** the approved KDF used.

The module must have a CAVP validation for any approved security functions used in any of the options.

The strength of the Data Protection Key is based on the strength of the Password and/or Passphrase used in key derivation. **SP 800-132** does not impose any strictly defined requirements on the strength of a password. It says that “passwords **should** be strong enough so that it is infeasible for attackers to get access by guessing a password.” Therefore, the vendor **shall** document in the module’s Security Policy the length of a password/passphrase used in key derivation and establish an upper bound for the probability of having this parameter guessed at random. This probability **shall** take into account not only the length of the password/passphrase, but also the difficulty of guessing it. The decision on the minimum length of a password used for key derivation is the vendor’s, but the vendor **shall** at a minimum informally justify the decision.

The iteration count determines the number of times the PRF is run per operation of the KDF. **SP 800-132** provides guidance on the lower limit to this value but leaves the value up to implementation specific conditions. The vendor **shall** document in the module’s Security Policy, a justification for the iteration count value used. If multiple iteration count values are used, the vendor **shall** document the conditions that lead to the various values.

Further, the vendor **shall** indicate in the module’s Security Policy that keys derived from passwords, as shown in **SP 800-132**, may only be used in storage applications.

Annotation

Refer to the [Management Manual - Annex A](#) for annotation examples (**PBKDF**).

Additional Comments

While the wording in [IG D.G](#) specifically prohibits using password-based SSP establishment methods in an approved mode, this does not contradict the statements in **SP 800-132** and in this IG, since **SP 800-132** allows the derived keys to be used only for storage applications. The SSP establishment addressed in [IG D.G](#) shows how to establish a key used for protecting sensitive data that may leave the cryptographic module.

Annex E – Approved authentication mechanisms

Annex F – Approved non-invasive attack mitigation test metrics

Change Summary

New Guidance

- Initial release – September 21, 2020.

Modified Guidance

- None.
-

Mapping IGs of FIPS 140-3 to FIPS 140-2

FIPS 140-3 IG	FIPS 140-2 IG
2.3.A - Binding of Cryptographic Algorithm Validation Certificates	1.4 - Binding of Cryptographic Algorithm Validation Certificates
2.3.B - Sub-Chip Cryptographic Subsystems	1.20 - Sub-Chip Cryptographic Subsystems
2.3.C - Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI)	1.21 - Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI)
2.4.A - Definition and Use of a non-Approved Security Function	1.23 - Definition and Use of a non-Approved Security Function
2.4.B - Tracking the Component Validation List	G.20 - Tracking the Component Validation List
3.4.A - Trusted Channel	2.1 - Trusted Path
4.1.A - Authorised Roles	3.1 - Authorized Roles
4.4.A - Multi-Operator Authentication	3.4 - Multi-Operator Authentication
5.A - Non-Reconfigurable Memory Integrity Test	9.13 - Non-Reconfigurable Memory Integrity Test
7.3.A - Testing Tamper Evident Seals	5.2 - Testing Tamper Evident Seals
7.3.B - Hard Coating Test Methods (Level 3 and 4)	5.4 - Level 3: Hard Coating Test Methods
9.3.A - Entropy Caveats	7.14 - Entropy Caveats
9.5.A - SSP Establishment and SSP Entry and Output	7.7 - Key Establishment and Key Entry and Output
9.6.A - Acceptable Algorithms for Protecting Stored SSPs	7.16 - Acceptable Algorithms for Protecting Stored Keys and CSPs

9.7.A - Zeroization of One Time Programmable (OTP) Memory	7.17 - Zeroization of One Time Programmable (OTP) Memory
10.3.A - Cryptographic Algorithm Self-Test Requirements	9.4 - Known Answer Tests for Cryptographic Algorithms
10.3.B - Self-test for Embedded Cryptographic Algorithms	9.2 - Known Answer Test for Embedded Cryptographic Algorithms
C.A - Use of non-Approved Elliptic Curves	A.2 - Use of non-NIST-Recommended Asymmetric Key Sizes and Elliptic Curves
C.B - Validation Testing of Hash Algorithms and Higher Cryptographic Algorithm Using Hash Algorithms	A.1 - Validation Testing of SHS Algorithms and Higher Cryptographic Algorithm Using SHS Algorithms
C.C - The Use and the Testing Requirements for the Family of Functions defined in FIPS 202	A.11 - The Use and the Testing Requirements for the Family of Functions defined in FIPS 202
C.D - Use of a Truncated HMAC	A.8 - Use of a Truncated HMAC
C.E - Key Generation for RSA Signature Algorithm	7.12 - Key Generation for RSA Signature Algorithm
C.F - Approved Modulus Sizes for RSA Digital Signature for FIPS 186-4	A.14 - Approved Modulus Sizes for RSA Digital Signature and Other Approved Public Key Algorithms
C.G - SP 800-67rev2 Limit on the Number of Encryptions with the Same Triple-DES Key	A.13 - SP 800-67rev1 Transition
C.H - Key/IV Pair Uniqueness Requirements from SP 800-38D	A.5 - Key/IV Pair Uniqueness Requirements from SP 800-38D
C.I - XTS-AES Key Generation Requirements	A.9 - XTS-AES Key Generation Requirements
C.J - Requirements for Testing to SP 800-38G	A.10 - Requirements for Vendor Affirmation of SP 800-38G
D.A - Acceptable SSP Establishment Protocols	D.2 - Acceptable Key Establishment Protocols
D.B - Strength of SSP Establishment Methods	7.5 - Strength of Key Establishment Methods
D.C - References to the Support of Industry Protocols	D.11 - References to the Support of Industry Protocols
D.D - Elliptic Curves and the FFC Safe-Prime Groups in Support of Industry Protocols	D.13 - Elliptic Curves and the MODP Groups in Support of Industry Protocols
D.E - Assurance of the Validity of a Public Key for SSP establishment	D.3 - Assurance of the Validity of a Public Key for Key Establishment
D.F - Key Agreement Methods	D.8 - Key Agreement Methods
D.G - Key Transport Methods	D.9 - Key Transport Methods
D.H - Requirements for Vendor Affirmation to SP 800-133	D.12 - Requirements for Vendor Affirmation to SP 800-133
D.I - The Use of Post-Processing in Key Generation Methods	7.8 - The Use of Post-Processing in Key Generation Methods
D.J - Entropy Estimation and Compliance with SP 800-90B	7.18 - Entropy Estimation and Compliance with SP 800-90B
D.K - Interpretation of SP 800-90B Requirements	7.19 - Interpretation of SP 800-90B Requirements

D.L - Critical Security Parameters for the SP 800-90A DRBGs	14.5 - Critical Security Parameters for the SP 800-90 DRBGs
D.M - Using the SP 800-108 KDFs in an Approved Mode	7.10 - Using the SP 800-108 KDFs in FIPS Mode
D.N - SP 800-132 Password-Based Key Derivation for Storage Applications	D.6 - Requirements for Vendor Affirmation of SP 800-132

End of Document