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: April 18, 2025

Table of Contents

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

1.A Binding and Embedding Cryptographic Modules

Applicable Levels:	*All*
Original Publishing Date:	*July 26, 2024*
Effective Date:	*July 26, 2024*
Last Modified Date:	*July 26, 2024*
Relevant Assertions:	
Relevant Test Requirements:	
Relevant Vendor Requirements:	

Background

ISO/IEC 19790:2012 and **ISO/IEC 24759:2017** incorporate implicit or explicit references to bound or embedded scenarios, as shown with added underlines for readability in the following excerpts.

[05.05] discussion of software/firmware integrity test:

For software and firmware modules and the software or firmware component of a hybrid module (except for the software and firmware components within a disjoint hardware component of a hybrid module):

- A cryptographic mechanism using an approved integrity technique **shall [05.05]** be applied to all software and firmware components within the module's defined cryptographic boundary in one of the following ways:
    - by the cryptographic module itself; or
    - by another validated cryptographic module operating in an approved mode of operation.

The above citation of [05.05] implicitly refers to a scenario where the module under test is bound to "another validated cryptographic module".

[05.13] discussion of software / firmware load test:

If the software or firmware that is loaded is associated, bound, modifies, or is an executable requisite of the validated module, then the software/firmware load test is applicable and **shall [05.13]** be performed by the validated module with the following exceptions.

- The cryptographic module is a software module and the loaded software image is a complete image replacement or overlay of the validated module.
- The cryptographic module is a firmware module of physical Security Level 1 and the loaded firmware image is a complete image replacement or overlay of the validated module.
- The cryptographic module is a hybrid software module and the loaded software image is a complete image replacement or overlay of the disjoint software components.
- The cryptographic module is a hybrid firmware module of physical Security Level 1 and the loaded firmware image is a complete image replacement or overlay of the disjoint firmware components.

Note underneath **ISO/IEC 19790:2012 [02.18]**:

In addition to the disjoint software or firmware component(s), the hardware component can also include embedded software or firmware.

ISO/IEC 19790:2012 Section 7.9.7 states:

Except at security level 4, zeroisation of protected PSPs, encrypted CSPs, or CSPs otherwise physically or logically protected within an additional embedded validated module (meeting the requirements of this International Standard) is not required.

The following items within the CMVP Caveats webpage relate to bound or embedded modules:

> **When operated in approved mode with module [module name] validated to FIPS 140-3 under Cert. #xxxx operating in approved mode**
>
> *The module's validation is bound to another validated cryptographic module.*
>
> Example: A software cryptographic module which requires services from another validated software cryptographic module operating in the same operational environment. Application services are available from either module.
>
> **This module contains the embedded module [module name] validated to FIPS 140-3 under Cert. #xxxx operating in approved mode**
>
> *If the module incorporates an embedded validated cryptographic module.*
>
> Example 1: A software cryptographic module which is compiled with a privately linked validated software cryptographic module operating in the same operational environment. Application services are only available from the module indicated on the certificate.
>
> Example 2: A hardware cryptographic module which has embedded within its physical boundary a validated cryptographic module.

Bound and embedded modules are also mentioned in the FIPS 140-3 IG articles specified below.

> In Table 1 of IG 9.5.A:
>
> - CM Hardware to/from EXT CM Hardware via directly connected EXT Path (e.g. CM bound to another CM): MD / EE
> - CM Software to/from CM Software via TOEPP Path (e.g. CM bound to another CM): MD / EE
> - CM Software to/from CM Software via INT Path (CM embedded within CM): N/A
> - CM Hardware to/from INT CM Hardware via INT Path (CM embedded within CM): N/A
>
> In IG D.J:
>
> Any new validation submission of a cryptographic module that obtains its entropy from a previously-validated embedded module **shall** comply with **SP 800-90B**.

Question/Problem

What is an embedded module? What is a bound module? What is required when the IUT relies on another validated cryptographic module? Can a module be bound to or embed a validated FIPS 140-2 module?

Resolution

1. The binding/bound and embedding/embedded cases are to be defined in relation to the cryptographic module under validation, identified as the Implementation Under Test (IUT). The following diagrams illustrate the binding/bound and embedding/embedded scenarios, respectively.

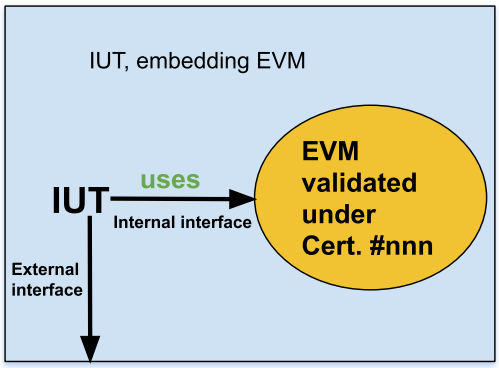


Figure 1: IUT (blue) as an Embedding Module

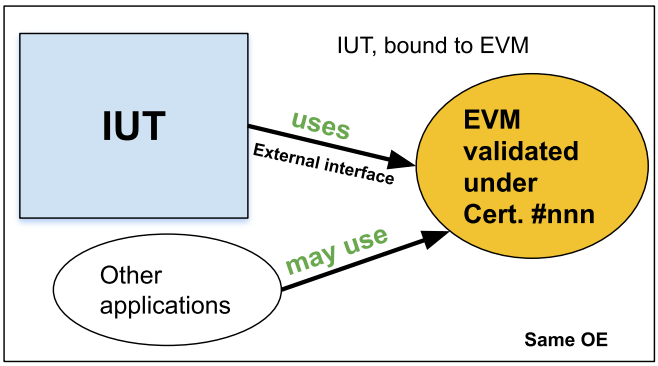


Figure 2: IUT (blue) as a Binding Module

Throughout this IG, the term IUT refers to the module undergoing validation. The term EVM refers to the existing validated module.

The primary benefit of this guidance is to be able to reuse an existing validated module to avoid repetitive testing and evaluation.

Embedding:

2. An embedded cryptographic module (as shown in Figure 1) is a module that is wholly contained within the module boundary of an IUT and whose services are provided solely to the IUT.

   Example: A hardware cryptographic module, the IUT, with EVM (hardware, firmware, software, or hybrid) embedded within its physical boundary.

Binding:

3. A binding cryptographic module (as shown in Figure 2) is a module with a separate, disjoint cryptographic boundary (IUT) that depends on an EVM. Application services are available from either cryptographic module.

   Example: A software cryptographic module (IUT) that requires services from a software EVM operating in the same operational environment (see Figure 2).

   The following **shall** apply to binding cases:

   a. In the case of software, firmware, and hybrid modules, the IUT and the EVM **shall** each operate on their tested operational environment(s) and those tested environments **shall** be the same (or one be within the other's).

   b. If the IUT relies on the EVM to meet one or more FIPS 140-3 requirement (i.e., does not meet all FIPS 140-3 requirements on its own), then the IUT submission **shall** demonstrate in the Test Report how the relied upon FIPS requirement(s) is met without being non-compliant to any other FIPS 140-3 requirement, considering IUT and EVM boundaries are independent / disjoint. E.g., **AS10.02** requires no external controls and no externally provided input test vectors ("external" to the module boundary), so the IUT may not be compliant to this requirement if it relied on the EVM for self-testing purposes.

   c. Per IG 9.5.A, SSPs moved or established between the binding IUT and bound EVM are subject to the **ISO/IEC 19790:2012** section 7.9.5 *Sensitive security parameter entry and output* and section 7.9.4 *Sensitive security parameter establishment* respectively because they are moved or established across the modules' respective cryptographic boundaries.

Embedding and Binding:

4. The following **shall** apply to both the embedding and binding cases:

   a. The EVM does not need to be (re)tested during the IUT validation. But the IUT, which utilizes the EVM, **shall** be fully tested.

b. For bound modules, the EVM **shall** be the same or higher security level as the IUT for all sections of the FIPS 140-3 standard, except for *Mitigation of other attacks* which can differ in security levels. This is to ensure the IUT provides the claimed security assurances despite utilizing the EVM to potentially meet certain FIPS 140-3 requirements and/or protect the IUT's sensitive data. E.g., if the IUT is security level 2 in *Roles, services, and authentication*, but the EVM is only security level 1, the EVM would not require authentication for the service(s) relied on by the IUT, which means the associated SSPs could potentially be modified/disclosed by an unauthenticated operator prior to the IUT request.

   For embedded modules, the EVM can be a lower security level as the IUT for all sections of the FIPS 140-3 standard, but if this is the case, the lab/vendor **shall** either: 1) demonstrate the EVM already meets the higher security level requirement(s); or 2) demonstrate that the IUT is able to meet the higher-level requirement(s) on behalf of the EVM.

c. The IUT **shall** inherit the Historical validation status if the EVM moves to the historical list for any reason (e.g., is sunset or an algorithm transition). The IUT could still become historical even if the EVM remains active if the IUT itself is subject to the historical list (e.g., an algorithm transition associated with an IUT algorithm). The vendors can revalidate their modules per FIPS 140-3 Management Manual Section 7.1 *Submission Scenarios*.

d. The EVM status **shall** be Active (i.e., is not on the Historical or Revoked list) at the time of the IUT submission to the CMVP. If the EVM becomes historical during IUT validation, the IUT will become historical once validation is completed.

e. To claim an EVM's implementation as an approved service / algorithm within the IUT documentation, the IUT **shall** only use EVM services / algorithms that are approved at the time of the IUT submission to the CMVP, even if the EVM implements algorithms in the approved mode that are no longer approved. E.g., the EVM may implement the transitioned **FIPS 186-4** DSA SigGen algorithm but still be on the Active list since the **FIPS 186-4** transition did not move the EVM to the historical list. The IUT could not claim the EVM's implemented DSA SigGen is approved if the IUT was submitted after the **FIPS 186-4** transition.

f. A FIPS 140-3 IUT cannot embed or bind to a FIPS 140-2 EVM.

Documentation Requirements:

5. The FIPS 140-3 IUT submission (e.g., Security Policy and validation test report) **shall** precisely identify the EVM by name, CMVP certificate number and version(s). The FIPS 140-3 Security Policy and validation test report for the IUT **shall** represent the functionality within its boundary with clear distinctions between IUT information and information associated with the EVM (e.g., clear separation of services, algorithms, SSPs, self-tests, zeroisation mechanisms, etc.). In some cases, the IUT might use only a subset of functionality provided by the EVM. The Security Policy and validation test report **shall** only cite the EVM functionality that is used by the IUT.

   This **shall** be done per below by marking "[EVM]" to the start of each applicable reference within the following WebCyptik tables (e.g., "[EVM] Perform Self-Tests" if the IUT calls the EVM's "Perform Self-Test" service):

CAVP Certs Table	Column to mark	EVM Marking
Algorithms	Category	[EVM]

MIS Table	Column to mark	EVM Marking
Table 6. Vendor Affirmed Algorithms	Algorithm Properties	[EVM]
Table 7. Non-Approved, Allowed Algorithms	Algorithm Capabilities	[EVM]
Table 8. Non-Approved Allowed Algorithms with No Security Claimed	Use/Function	[EVM]
Table 9. Non-Approved, Not Allowed Algorithms	Use/Function	[EVM]
Table 10. Security Function Implementations	Description	[EVM]
Table 11. Entropy Certificates	Cannot mark the table; must note in Security Policy after the table.	
Table 15. Roles	Name	[EVM]
Table 16. Approved Services		
Table 17. Non-Approved Services		
Table 24. SSPs	Name	[EVM]
Table 25. Pre-Operational Self-Tests	Details	[EVM]
Table 26. Conditional Self-Tests	Details	[EVM]
Table 27. Error States	Description	[EVM]

All other areas of the Security Policy where EVM information is needed that are not shown in the above tables, **shall** also be marked with [EVM].

[**AS02.24**] Each IUT service **shall** indicate use of approved algorithms even when it requests the EVM to use any approved algorithms on its behalf (see IG 2.4.C).

Additional Comments

1. In the case of software, firmware, and hybrid modules, since the IUT's tested OE must match (or be a subset of) the tested OEs of the EVM, a post-validation addition to the IUT's tested OEs would require an addition to the tested OEs of the EVM if the newly added OEs are not a subset of the OEs of the EVM (see the FIPS 140-3 Management Manual 7.1.7 OEUP).
2. If the EVM supplies the IUT with entropy and has an "ENT" validation (as opposed to an ESV), the IUT will show the "ENT" on its certificate.

Section 2 – Cryptographic module specification

2.3.A Binding of Cryptographic Algorithm Validation Certificates

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

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 Table 3 in MIS Table Descriptions).
    - 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	March 26, 2024
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 19790:2012** 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 19790:2012** 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 available 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. It is not required for the tester to manipulate these components as required by TE02.13.03 unless the vendor has identified an operational security concern. Additionally, the lab **shall** elaborate on the reasonability of the vendor's rationale in TE02.13.03. 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 of Section 7.9 (*Sensitive security parameter management*), 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 HMI and an intervening functional subsystem for Security Levels 1 and 2 on the same single chip does not require meeting the sensitive security parameter establishment requirements, 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 modules that are Security Levels 3 or 4, SSP establishment is AD / 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. **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 for Section 7.9 (*Sensitive security parameter management*) is considered not having SSP establishment across their sub-chip cryptographic subsystem boundary per IG 9.5.A.

   - For Security Level 3 and Security Level 4 modules for (Section 7.9 *Sensitive security parameter management*), CSP establishment is AD / 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 UPDT or a FS.

   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 PTSC.

   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 UPDT or a FS.

Additional Comments

1. 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.
2. 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.
3. After validation, if there are any changes to versioning within the single chip (security relevant or not) even if solely to the components outside of the sub-chip cryptographic subsystem boundary, the module **shall** be validated or re-validated to maintain validated status (e.g., see the FIPS 140-3 MM Section 7.1.9).

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

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	October 23, 2024
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? What are the testing requirements when a module supports PAA and/or PAI?

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.

Hybrid Software/Firmware 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 either a Hybrid Software Module embodiment, or a Hybrid-Firmware module embodiment.

PAA

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

PAI

- **Module versioning** information **shall** include the part number or version of the processor chip.
- **Operational Environment:** <OS> running on <platform> with <processor> 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 either a Software module embodiment or a 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:**
  - **<**OS> running on <platform> with <processor> with PAA;
  - <OS> running on <platform> with <processor> without PAA

PAI

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

Testing requirements

A module is considered a hybrid if *all* the Operational Environments (OEs) *only* support *with* PAA/PAI and were tested as such.

A module can support PAA/PAI and be considered a software/firmware module instead of a hybrid if one of the following two options are chosen:

1. Perform module testing *with* and *without* PAA/PAI for all OEs on the certificate.
   a. OEs would either be listed *with* and *without* on the certificate (which is typical), or *with* or *without* (rare) if the algorithm and operational testing is performed on both options and shown in the Test Report. Sometimes a vendor may choose this second option as to encourage a user in using a PAA/PAI or non-PAA/PAI solution (even though both are technically validated).
2. Perform module testing *without* PAA/PAI for at least one OE on the certificate, and *with* PAA/PAI for at least one OE on the certificate. Testing *with* and *without* may be on the same OE or may be on another, "similar", OE following the rules below.
   a. The module **shall** be tested *with* PAA/PAI on *every* OE on the certificate that supports a PAA/PAI. This is to confirm every PAA/PAI implementation/combination is tested, since there may be subtle differences in both the code and the PAA/PAI in the processor.
   b. For every OE that is tested *with* PAA/PAI, there **shall** be another "similar" or identical OE tested *without* PAA. Examples of OEs that may or may not be "similar" are provided in IG 2.3.A Resolution #2 (e.g., a 64-bit OE is not "similar" to a 32 bit-OE, even if everything else is identical). The testing lab **shall** justify the "similar" claim in the Test Report. It is recommended that "similar" requests be submitted to the CMVP via the existing Request for Guidance process detailed in the Management Manual Section 2.5 *Request for Guidance from CMVP*. It is at the discretion of the CMVP to accept this claim.
   c. There **shall** be no differences in module binaries executed on any of the OEs that are considered "similar", whether testing *with* or *without* the PAA/PAI (note: the PAA/PAI code is considered outside of the logical boundary of the module). E.g., the module has identical binaries running on all "similar" OEs, and this code was tested in at least one instance of both *with* and *without* PAA/PAI.
   d. It is acceptable if some of the OEs only support *without* PAA/PAI and not *with* PAA/PAI – as this is still considered a software/firmware module. E.g., OE1 uses an Intel Xeon E5-2600 and was tested *with* and *without* PAA, while OE2 uses an ARMv5 that does not implement a PAA/PAI and was only tested *without* PAA/PAI.

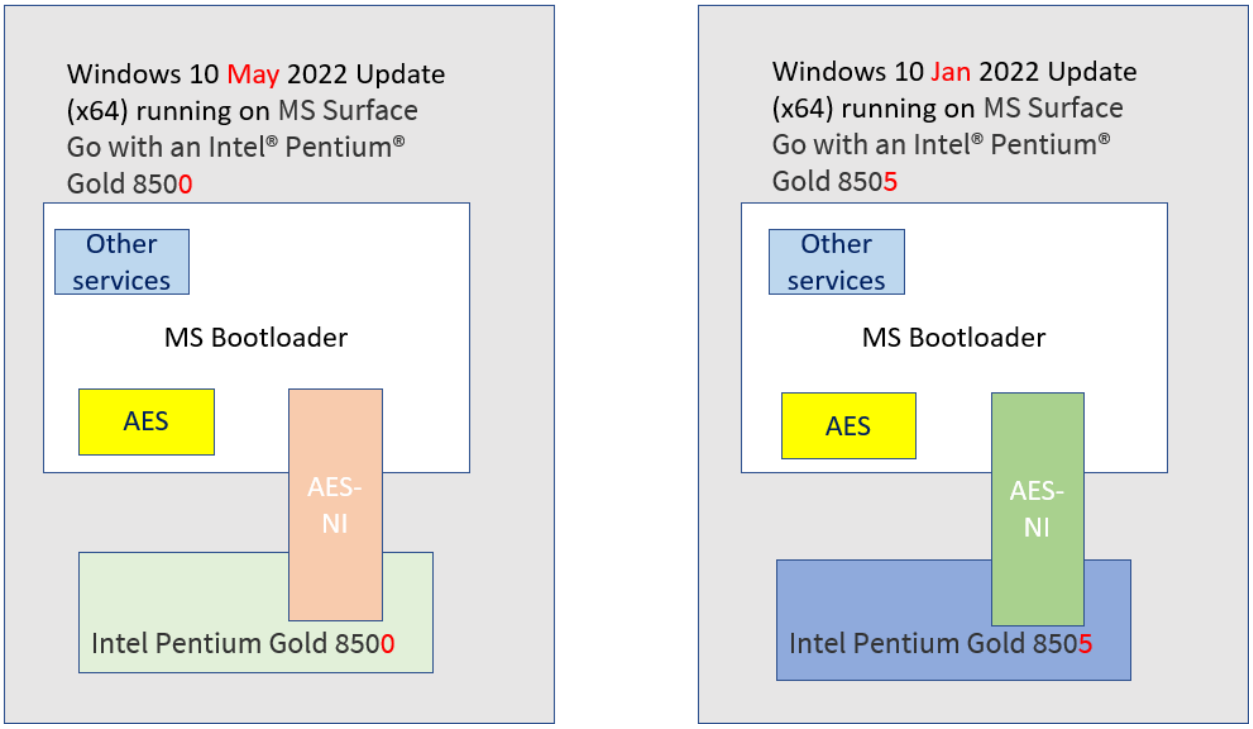
e. It is acceptable if some of the OEs were only tested *with* PAA/PAI and not *without* PAA/PAI – as this is still considered a software module since that code branch (*without* PAA/PAI) has already been tested on a "similar" OE. E.g., OE1 uses OS1 on an Intel Xeon E5-2600 and was only tested *with* PAA, while OE2 uses OS1 on an Intel Xeon E5-2650 that was tested *with* and *without* PAA.

f. The module certificate **shall** never include an OE that was not individually tested by the lab.

g. The Test Report **shall** include evidence demonstrating all module/PAA/PAI code branches used by OEs on the certificate are tested within the combination of the tested OEs.

h. The above rules apply only for MODULE LEVEL testing, and does not address CAVP testing (e.g., see CAVP FAQ AES.2).

Below is a diagram to help explain option 2.



- The diagram shows a module running on two slightly different environments. The module was tested *without* PAA only once, since the module binary that includes the AES implementation *without* PAA(Yellow box) is exactly the same in both environments. Thus, there is assurance that the AES *without* PAA will function on any "similar" environment.
- However, the module was tested *with* PAA twice, since the processor is different in both environments, and testing on both guarantees that all code is exercised, including *with* the PAA code on the processor itself (peach and green boxes).
- The module certificate can only claim what is tested (i.e., *with* PAA on both environments, and *without* PAA on one).

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, Sunny Cove, or Golden Cove micro-architectures: PAA = AES-NI

- Accelerator sub-functions for AES implementations

- Intel Processors – Atom, Celeron, Xeon, and Pentium with Goldmont, Goldmont Plus, Sunny Cove, Golden 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), TDES, SHA-1, SHA2

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. If only part of a single chip is to be included in a submission for its cryptographic capabilities, it is not necessarily or automatically a PAA/PAI. A hardware sub-chip module per IG 2.3.B should be considered.
4. Please contact the CMVP to request a new PAA or PAI implementation be added to the "known" list (could be part of the module submission or as an RFG). Requests **shall** include supporting evidence (public and/or private) that justifies the PAA or PAI implementation, with independent summarization/verification/confirmation by the CSTL. CAVP testing is also required. For PAI only, HDL is required which additionally requires CSTL review/verification of the HDL.
5. If the PAI security function appears on the list of known PAIs, its HDL is not required for validation of modules using it.
6. For CAVP and ESV certificates, testing with and without PAA/PAI is not required if the algorithm or entropy source does not utilize the corresponding PAA/PAI.
7. PAA/PAI, as the name indicates, are specific to a processor and cannot be any hardware cryptographic implementations. PAA/PAI execute when requested by the processor and/or via specific processor instructions.

2.3.D Excluded Components

Applicable Levels:	All
Original Publishing Date:	March 17, 2023
Effective Date:	March 17, 2023
Last Modified Date:	March 17, 2023
Relevant Assertions:	AS02.13
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

ISO/IEC 19790:2012 Section 7.2.3.1:

Hardware, software and/or firmware components within the cryptographic boundary may be excluded from the requirements of this International Standard. The excluded hardware, software or firmware components **shall [02.13]** be implemented in a manner to not interfere or compromise the approved secure operation of the cryptographic module.

ISO/IEC 24759:2017:

AS02.13: (Specification — Levels 1, 2, 3, and 4) The excluded hardware, software or firmware components shall be implemented in a manner to not interfere or compromise the approved secure operation of the cryptographic module.

VE02.13.01: The vendor shall describe the excluded components of the module and justify that these components will not interfere with the approved secure operation of the module.

VE02.13.02: The vendor documentation shall provide the rationale for excluding each of the components. The rationale shall describe how each excluded component, when working properly or when it malfunctions, cannot interfere with the approved secure operation of the module. Rationale that may be acceptable, if adequately supported by documentation, includes the following.

a) The component is not connected with security relevant components of the module that would allow inappropriate transfer of SSPs, plaintext data, or other information that could interfere with the approved secure operation of the module.

b) All information processed by the component is strictly for internal use of the module, and does not in any way impact the correctness of control, status or data outputs.

TE02.13.01: The tester shall verify from the vendor provided documentation that the excluded components of the cryptographic boundary will not interfere with the approved secure operation of the module.

TE02.13.02: The tester shall verify the correctness of any rationale for exclusion 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.

TE02.13.03: The tester shall manipulate (e.g. to cause the component to operate not as designed) the excluded components in a manner to cause incorrect operation of the excluded component. The tester shall verify that the incorrect operation of the excluded component shall not interfere with the approved secure operation of the module.

Question/Problem

What are the testing requirements to meet TE02.13.03?

Resolution

It is up to the testing lab and the vendor to come up with the appropriate testing approach to meet TE02.13.03. The testing approach and rationale **shall** be documented in this TE. Testing may rely on code review and/or documentation alone if and only if behavioral (e.g., using a debugger, code manipulator/injector, simulator, or

other tool to manipulate data and/or characteristics of the component that may impact the behavior of an excluded component) and/or physical (e.g., shorting/removing pins, voltage manipulation) methods to cause the incorrect operation of the excluded component are infeasible or impractical for a given module. Otherwise, behavioral and/or physical methods are required. Testing is considered infeasible or impractical when such manipulations are understood to permanently damage the module, or the system in which it is contained, without exposing any security relevant data or achieving any desired security goals beyond simply rendering the entire module inoperable. Tester **shall** verify rationale provided by the vendor in such a scenario.

Testing excluded components may be at a high-level (e.g., the entire memory component of a module, the power supply, line cards, fans, etc.) and/or at a low-level (e.g., the underlying individual hardware components such as switches, resistors, capacitors, or inductors), depending on which test is most appropriate based on the analysis performed by the lab and vendor.

The tests **shall** focus on the secure operation of the module, but not necessarily on other aspects such as reliability and/or quality, unless they may help prove lack of interference with the approved secure operation of the module.

Acceptance of such testing is at the discretion of the CMVP. Labs can submit their proposed test approach to the CMVP for pre-approval via an RFG (see Management Manual Section 2.5 *Request for Guidance from CMVP*).

Additional Comments

1. After validation, if an excluded component is modified, the module **shall** be validated or re-validated to maintain validated status, as excluded components are still considered within the boundary of the module. In such case, during validation or revalidation, compliance to **AS02.13** is required.
2. The details of the excluded components (including their versioning) **shall** be captured in TE02.14.01.
3. The overall module version **shall** change if excluded components are changed.

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

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	April 18, 2025
Relevant Assertions:	AS02.21
Relevant Test Requirements:	TE02.21.01 and TE02.21.02
Relevant Vendor Requirements:	VE02.21.01 and VE02.21.02

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 unauthorised operation or use;
- prevent the unauthorised disclosure of the contents of the cryptographic module, including CSPs;
- prevent the unauthorised and undetected modification of the cryptographic module and cryptographic algorithms, including the unauthorised 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 Security Policy's list of non-approved algorithms allowed in the approved mode of operation with no security claimed (see **SP 800-140B**). However, the module's certificate **shall not** include these algorithms as they do not claim any security and are not used to meet any requirement of FIPS 140-3.

A non-approved cryptographic algorithm **shall not** share the same keys or CSPs that are 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 with no security claimed

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 (note, the XOR itself is not a cryptographic algorithm but rather a part of a cryptographic algorithm that may include the establishment of the secret value for example)
   - 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 zeroisation 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 [1] in the TLS 1.0 / 1.1 KDF

      SP 800-135rev1 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.

[1] May be allowed in an approved mode of operation when used as part of an approved key transport scheme (e.g., SSL v3.1) where no security is provided by the algorithm.

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 <u>not allowed in any mode</u>

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 (Triple-DES encryption is a disallowed algorithm and is only provided here for illustrative purposes) key K = (K1, K2, K3) which is a CSP, as it may be used by an approved Triple-DES algorithm. The value E = DES_{K1} (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 186-5**, 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

1. 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. In addition, attempts to make use of this IG to include algorithms in the approved mode will not be accepted unless all of the following are met: 1) the algorithm is not used whatsoever to meet any FIPS 140-3 requirements; 2) the algorithm does not access or share CSPs in a way that counters the requirements of this IG; 3) the algorithm is either: i) not intended to be used as a security function (e.g. interoperability or for memory wear leveling); ii) redundant to an approved algorithm (e.g. double encryption); iii) a cryptographic or mathematical operation applied for "good measure" but not for providing sound security (e.g. XORing a CSP with a secret value, using a proprietary algorithm, or using non-approved algorithms to obfuscate stored CSPs which are considered plaintext); 4) the algorithm's non-approved use and purpose (from 3) above) is unambiguous to the operator and can't be easily confused for a security function.

   A key point to consider for 4) above is the purpose or use of the service that utilizes the non-approved algorithms. For example, if the purpose is solely to provide cryptographic functionality to a calling application, then the service is easily confused for a security function (e.g., a software library that provides non-approved key agreement, key transport, or general encryption/decryption services) and the algorithm(s) within this service cannot claim no security per this IG. But if it is clear that the purpose of a service does not include providing any security functionality recognized in FIPS 140-3 then the service can use non-approved algorithms in the approved mode if provisions of this IG are met. Examples of such services are status output (possibly, using the non-security provisions of the SNMP protocol) and obfuscating CSPs for internal storage that is considered plaintext.

2.4.B Tracking the Component Validation List

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	July 26, 2024
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

Background

In response to vendor and user requirements, the CAVP and CMVP have identified several components of the approved algorithms that can be used in an approved mode of operation. These components are included in this IG and labeled "CVL" (Component Validation List) in the CMVP documentation.

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

What are the available testable components of the approved algorithms? Which documents specify the functions that each of these components performs? What are their usage restrictions, if any? How do they map to the CAVP ACVTS certificate? Can an algorithm component be vendor affirmed?

Resolution

The following components can be tested and will be documented as "(CVL)" in the CAVP Certs Table of the module validation. TE02.20.01 **shall** explain how each CVL meets this IG and is used in an approved manner by the module.

1. An ECC CDH primitive from section 5.7.1.2 of **SP 800-56Arev3**. The test performs the multiplication of a point on a NIST-recommended elliptic curve by an integer and verifies that the x-coordinate of the resulting point has been computed correctly. The integer in question is defined in **SP 800-56Arev3** as the product of the tested party's private key d_A and the curve's co-factor h.

   Usage Restriction:

   Shall only be used within the context of a **SP 800-56Arev3** KAS (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the KAS-ECC CDH-Component (CVL) shall only be used within the context of an **SP 800-56Arev3** KAS").

   CAVP certificate and denotation:

   Within the context of the CAVP ACVTS, this component test is identified by the following algorithm/mode/revision capability combination:

   - "algorithm": "KAS-ECC",
   - "mode": "CDH-Component",
   - "revision": "Sp800-56Ar3".

   This testing is denoted by "KAS-ECC CDH-Component SP800-56Ar3" on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search.

2. An RSA signature generation per **FIPS 186-5** without the computation of a hash which is presumed to have already been computed.

   This RSA test verifies the ability of a module to perform the RSASP1 signature primitive described in PKCS#1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002, Section 5.2.1. RSASP1 takes as input an RSA private key and a message digest and outputs a signature representative or an error message. The test 1) verifies the ability of the module to detect an invalid message representative and 2), given a valid message representative, the ability of the module to compute the correct signature representative. The private key can be in the form of 1) a pair (n, d) or 2) a quintuple (p, q, dP, dQ, qInv). If the private key takes the form of a pair (n, d), then the signature representative is computed as described in Section 5.2.1 Step 2) a. of the standard. If the private key takes the form of a quintuple (p, q, dP, dQ, qInv), then the signature representative is computed as described in Section 5.2.1 Step 2) b. of the standard.

   As with the full RSA signature generation, defined in **FIPS 186-5** and further explained in IG C.F, a component of an RSA signature generation, often referred to as RSASP1, can be used with any RSA modulus *n* (or a secret RSA key *d*) no smaller than 2048-bit long. The RSASP1 component implementations **shall** be tested by an accredited testing lab for all implemented RSA modulus lengths where the CAVP testing is available. If there is no CAVP testing for an RSASP1 component with a particular RSA modulus length then such modulus size does not need to be tested; however, at least one modulus length for which a test exists **shall** be implemented and tested by the CAVP. This test supports the 2048, 3072 and 4096-bit moduli.

   Usage Restriction:

   Shall only be used within the context of a **FIPS 186-5** signature generation (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the RSA SigGen (CVL) shall only be used within the context of a **FIPS 186-5** signature generation").

   CAVP certificate and denotation:

   Within the context of the CAVP ACVTS, the RSA signature generation component test is identified by the following algorithm/mode/revision capability combination:

   - "algorithm": "RSA",
   - "mode": "signaturePrimitive",
   - "revision": "1.0".

   This testing is denoted by "RSA Signature Primitive" on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search.

3. An ECDSA signature generation per **FIPS 186-5** without the computation of a hash which is presumed to have already been computed.

   This ECDSA signature generation component 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 digest is supplied to the test. See Section 7.3 of ANS X9.62-2005 or Section 6.4.1 of **FIPS 186-5** for more information.

   Usage Restriction:

   Shall only be used within the context of a **FIPS 186-5** signature generation (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the ECDSA SigGen (CVL) shall only be used within the context of a **FIPS 186-5** signature generation").

   CAVP certificate and denotation:

   Within the context of the CAVP ACVTS, this component test is identified by any of the following algorithm/mode/revision/componentTest capability combinations:

- "algorithm": "ECDSA",
- "mode": "sigGen",
- "revision": "1.0",
- "componentTest": true;

or

- "algorithm": "ECDSA",
- "mode": "sigGen",
- "revision": "FIPS186-5",
- "componentTest": true;

or

- "algorithm": "detECDSA",
- "mode": "sigGen",
- "revision": " FIPS186-5",
- "componentTest": true.

This testing is denoted by "ECDSA SigGen (FIPS186-4) [2]", "ECDSA SigGen (FIPS186-5)" or "Deterministic ECDSA SigGen (FIPS186-5)" accompanied by "Component" listed in the details field on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search

4. An RSA decryption operation using an exponentiation for key encapsulation, as specified in section 7.1.2.1 of **SP 800-56Brev2** published in March 2019. RSADP is used as shorthand to refer to the RSA decryption primitive in **SP 800-56Brev2**. This test only supports the 2048-bit modulo.

   Usage Restriction:

   Shall only be used within the context of a **SP 800-56Brev2** KTS (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the RSA Decryption Primitive (CVL) shall only be used within the context of a **SP 800-56Brev2** KTS").

   CAVP certificate and denotation:

   Within the context of the CAVP ACVTS, this component test is identified by the following algorithm/mode/revision capability combination:

   - "algorithm": "RSA",
   - "mode": "decryptionPrimitive",
   - "revision": "1.0".

   This testing is denoted by "RSA Decryption Primitive" on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search

5. An RSA decryption operation using an exponentiation for key encapsulation, as specified in sections 7.1.2.1 and 7.1.2.2 of **SP 800-56Brev2** published in March 2019, or using the process specified in section 7.1.2.3 of the same publication where the private key is in the Chinese Remainder Theorem (CRT) format. RSADP is used as shorthand to refer to the RSA decryption primitive in **SP 800-56Brev2**. This test supports the 2048, 3072 and 4096-bit moduli.

   Usage Restriction:

[2] This test is mathematically equivalent to the comparable FIPS 186-5 test, and therefore, a vendor can claim compliance to FIPS 186-5 using this FIPS 186-4 test. However, all new testing is expected to use the FIPS 186-5 test.

Shall only be used within the context of a **SP 800-56Brev2** KTS (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the RSA Decryption Primitive SP800-56Brev2 (CVL) shall only be used within the context of a **SP 800-56Brev2** KTS").

CAVP certificate and denotation:

Within the context of the CAVP ACVTS, this component test is identified by the following algorithm/mode/revision capability combination:

- "algorithm": "RSA",
- "mode": "decryptionPrimitive",
- "revision": "Sp800-56Br2".

This testing is denoted by "RSA Decryption Primitive SP800-56Br2" on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search

6. The key derivation functions from the following protocols and standards documented in **SP 800-135rev1**: IKEv1, IKEv2, TLS 1.0, 1.1 and 1.2, SSHv2, SRTP (using the 32-bit or 48-bit index value in SRTCP), SNMPv3, TPMv1.2, ANSI X9.63-2001 KDF and ANSI X9.42-2001 KDF.

   Usage Restriction:

   Shall only be used in the context of their respective protocols (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the KDF SSH (CVL) shall only be used within the context of the SSHv2 protocol").

   CAVP certificate and denotation:

   Within the context of the CAVP ACVTS, these component tests are identified by the following algorithm/mode/revision capability combinations:

   - "algorithm": "kdf-components",
   - "revision": "1.0",
   - "mode": "ikev1" | "ikev2" | "tls" | "ssh" | "srtp" | "snmp" | "tpm" | "ansix9.63" | "ansix9.42";

   or

   - "algorithm": "TLS-v1.2",
   - "mode": "KDF",
   - "revision": "RFC7627".

   These tests are denoted by "KDF IKEv1", "KDF IKEv2", "KDF TLS", "TLS v1.2 KDF RFC7627", "KDF SSH", "KDF SRTP", "KDF SNMP", "KDF TPM", "KDF ANS 9.63" and "KDF ANS 9.42" on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search.

7. 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**.

   Usage Restriction:

   Shall only be used in the context of the TLS 1.3 protocol (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the TLS v1.3 KDF (CVL) shall only be used within the context of the TLS 1.3 protocol").

   CAVP certificate and denotation:

Within the context of the CAVP ACVTS, this component test is identified by the following algorithm/mode/revision capability combination:

- "algorithm": "TLS-v1.3",
- "mode": "KDF",
- "revision": "RFC8446".

This testing is denoted by "TLS v1.3 KDF" on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search.

8. 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 800-56Arev3** and Sections 5.6, 8.2.3, 8.3.3 and 9.2.4 of **SP 800-56Brev2**. 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.

   Usage Restriction:

   Shall only be used within the context of a **SP 800-56Arev3** or **SP 800-56Brev2** KAS (enforced by the module in that this CVL is not used internally for other purposes; but if external / operator-accessible enforced by operator guidance within the Security Policy, e.g., "the KAS-KC SP800-56 (CVL) shall only be used within the context of an **SP 800-56Arev3** or **SP 800-56Brev2** KAS").

   CAVP certificate and denotation:

   Within the context of the CAVP ACVTS, this component test is identified by the following algorithm/revision capability combination:

   - "algorithm": "KAS-KC",
   - "revision": "Sp800-56".

   This testing is denoted by "KAS-KC SP800-56" on the CAVP webpage: https://csrc.nist.gov/projects/cryptographic-algorithm-validation-program/validation-search.

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.

For modules submitted to the CMVP prior to June 30, 2023 (per the CMVP Programmatic Transitions page), the following component can be vendor affirmed and documented as a CVL in the *Vendor Affirmed Algorithms* table in WebCryptik.

1. The key derivation function from the following protocol and standard documented in **SP 800-135rev1**: SRTP (using the 48-bit index value in SRTCP per the NIST published Informative Note). The Test Report **shall** include documentation demonstrating the module's compliance to **SP 800-135rev1** Section 5.3. The CMVP has transitioned away from vendor affirming the SRTP KDF using the 48-bit index value in SRTCP since the CAVP has developed a test for it as of March 21, 2023.

The Security Policy **shall** individually list the vendor affirmed components that may be called during the operation of the module.

Additional Comments

1. The testing of compliance to **SP 800-56Arev3** 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 800-56Arev3** further shows how to apply the key derivation functions defined in **SP 800-56Crev2**, the computation of a shared secret is viewed as a core functionality defined in **SP 800-56Arev3**. 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 (see IG D.F for more information on KAS-SSC).
2. The details of the CAVP component testing are provided at https://pages.nist.gov/ACVP/#supported by searching for the relevant tests.

3. Refer to IG 10.3.A for the applicability of self-tests to the tested components that have been issued the CVL certificates.

2.4.C Approved Security Service Indicator

Applicable Levels:	All
Original Publishing Date:	May 4, 2021
Effective Date:	May 4, 2021
Last Modified Date:	November 22, 2023
Relevant Assertions:	AS02.24
Relevant Test Requirements:	TE02.24.01-02
Relevant Vendor Requirements:	VE02.24.01

Background

ISO/IEC 19790:2012 section 7.2.4.2 states:
All services **shall [02.24]** provide an indicator when the service utilizes an approved cryptographic algorithm, security function or process in an approved manner and those services or processes specified in 7.4.3.

ISO/IEC 24759:2017 has the following requirement:

AS02.24: (Specification — Levels 1, 2, 3, and 4)

All services shall provide an indicator when the service utilizes an approved cryptographic algorithm, security function or process in an approved manner and those services or processes specified in *{ISO/IEC 19790:2012}* 7.4.3.

Required Vendor Information

VE02.24.01: The vendor provided documentation shall specify the indicator for each service.

Required Test Procedures

TE02.24.01: The tester shall verify that the vendor provided documentation contains a description of the indicator when the service utilizes an approved cryptographic algorithm, security function or process in an approved manner.

TE02.24.02: The tester shall execute all services and verify that the indicator provides an unambiguous indication of whether the service utilizes an approved cryptographic algorithm, security function or process in an approved manner or not.

Question/Problem

What services need an indicator? **AS02.24** mentions services or processes specified in **ISO/IEC 19790:2012** 7.4.3. Nevertheless, TE02.24.01 and TE02.24.02 do not include services or processes specified in section 7.4.3. Section 7.4.3 requires a cryptographic module to provide services such as show module's versioning information, show status and pre-operational self-tests. Should there be an indicator showing a module's versioning information or show status service, when these services can run in both approved and non-approved modes of operation? How can a pre-operational self-test be indicated by a cryptographic module even before the module becomes functional?

What is the expected level of granularity for the indicator? Is it that a module shall indicate when it is using an approved service? Or, is it that each service shall indicate when it is using an approved cryptographic algorithm? Can a description in the module's security policy be a service indicator? Can the approved mode indicator under FIPS 140-2 DTR **AS01.04** be the service indicator under FIPS 140-3? When should the service indicator be available to the user?

Resolution

Per the definition provided in **ISO/IEC 19790:2012** Section 3.111, a service is any externally operator invoked operation and/or function that can be performed by a cryptographic module. A service corresponds to a specific task or callable function to be performed by the module. Services provided by a module may not have one-to-one correspondence to the API functions implemented by the module. A service (e.g. Random Number Generation) may invoke a group of API functions. On the other hand, an API function may provide different services (e.g. symmetric encryption vs. asymmetric encryption) depending on the different values of

some or all of its input parameters. A vendor may choose to document services in terms of API functions, if appropriate. Nevertheless, API functions are not required to be the only way to specify services.

ISO/IEC 19790:2012 Section 7.2.4.2 does mention "approved and non-approved services", but it doesn't define them. IG 2.4.A defines an approved security function as a 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]. This definition can naturally be extended to state that approved security services are those that utilize the approved security functions in an approved manner.

Services provided by a cryptographic module can be categorized into the following groups:

1. services that use approved (including allowed) security functions or processes in an approved manner,
2. services that do not use any security functions (i.e. approved or non-approved), but are described in **ISO/IEC 19790:2012** 7.4.3 (e.g. show module's versioning information, show status),
3. services that use non-approved algorithms but do not claim security as specified in IG 2.4.A and are allowed in an approved mode of operation,
4. services that use non-approved and not allowed security functions or processes, and therefore not available for use in an approved mode of operation,
5. services that may perform non-security actions in either approved or non-approved mode of operation.

Group 1 services represent approved security services, but they are not the only services run in the approved mode of operation. Services in groups 2, 3 and 5 may also run in that mode. Services in all groups except for group 4 are approved (including allowed) in the sense that they are approved (or allowed) to run in an approved mode of operation.

Unlike FIPS 140-2 that requires an indication of approved mode of operation, **ISO/IEC 19790:2012** is intended to indicate the approved security services, which is a more granular and security relevant indication than its broader counterpart: the indication of an approved mode of operation.

This IG clarifies **AS02.24** by interpreting the following:

- TE02.24.01 and TE02.24.02 focus only on services utilizing cryptographic algorithms, security functions or processes to determine if their implementation is approved or not but does not discuss other services or processes specified in **ISO/IEC 19790:2012** 7.4.3 which are mentioned in **AS02.24**.
- **AS02.24** mentions services utilizing approved security functions twice: once where it is explicitly stated and once where it is implicitly stated as "those services or processes specified in **ISO/IEC 19790:2012** 7.4.3" since "Perform approved security functions" is defined in **ISO/IEC 19790:2012** 7.4.3.
- **AS02.24** seems to require an indicator for the services when they utilize "those services or processes specified in **ISO/IEC 19790:2012** 7.4.3". However, this IG interprets this as the services specified in **ISO/IEC 19790:2012** 7.4.3 themselves do not need an indicator, but rather that the services which utilize services specified in **ISO/IEC 19790:2012** 7.4.3 need an indicator. As an example, consider that self-tests do not need an indicator, but a service providing on-demand self-tests do need an indicator. If on-demand self-tests are performed through power-cycling, resetting or rebooting without a purposefully designed service utilizing the self-tests, then no indicator is needed besides that required in **AS10.08** and **AS10.11**. For the services utilizing zeroisation function(s) to zeroise the parameters as specified in **ISO/IEC 19790:2012** 7.9.7, IG 9.7.B addresses the requirement.

This IG makes an important distinction between approved security services and services provided in the approved mode of operation (i.e. approved services for convenience). Based upon this distinction, this IG provides clarity to **AS02.24** by requiring all approved security services to have an indicator and only those services need to have an indicator. For example, the services in Group 2 above do not need a service indicator unless used as part of another approved service that needs this indicator (e.g. a service that provides on-demand self-tests or a service that performs approved security functions).

An indicator is a means of communicating a specific status. So, a service indicator is a means of communicating a specific status to a User (e.g. human operator, application or another module) as it relates to approved and non-approved security functions or a process identified by the Vendor as performed by a cryptographic module.

Whether a security function is approved may be context sensitive to the service in which it is included. For example, at Overall Security Rating 1, the SHA-1 function is an approved algorithm if it is used for an integrity check service, but it is not approved if it is used as part of a digital signature generation service. Therefore, it is not required to provide the indicator at the API level of cryptographic functions, as long as the service implementing the API provides the corresponding indicator that unambiguously indicates the approved security services.

If the module operator (e.g., calling application) can do things outside of the module's control/visibility that can take an otherwise approved algorithm and use it in a non-approved way (e.g., use PBKDF and/or AES XTS outside of storage applications), the corresponding module service may still be considered approved (and if so, **shall** have an approved indicator per **AS02.24**) and the Security Policy **shall** clarify how to use the service in an approved manner (per ISO 19790 B.2.2 on *Overall security design and the rules of operation*).

A service indicator allows the operator of the module to unambiguously determine if and when an approved security service is in use. The indicator does not need to be physical or human readable only. It can be any indication that is externally accessible from the status interface provided by the module and verifiable by the operator of the module.

For all approved security services, a global module-level indicator, including the approved mode indicator under FIPS 140-2 DTR AS01.04, is acceptable if the module operator can unambiguously determine an approved security service is in use. If a module can provide an approved and non-approved security service concurrently, or a module does not require configuration of all possible approved services for an approved mode, then a global module-level indicator that is not able to unambiguously indicate the approve security service does not meet the requirement.

The Security Policy **shall** provide a complete list of all approved and non-approved services along with details on each service and their respective indicators (if applicable). The security policy may be used to provide interpretation for the indicator(s) provided by the module, but the description in the security policy alone does not fulfill the requirement.

The following are some example scenarios showing how an indicator may be provided by a module. *These serve as examples only*. A vendor may design and implement an indicator differently to meet **AS02.24** and related VEs and TEs.

1) A separate indicator for each of the approved security services implemented by the module. This can be done through return code, log message etc. for each service call. When a service consists of more than one API function, the service indicator can be provided by the API function that, when invoked, can determine whether the service is approved or not (e.g. in a Digital Signature Generation service, the module will be able to determine whether the service is approved only once the signature generation algorithm, hashing algorithm, and key size are provided).

2) A global indicator for the module that only supports approved services in an approved manner. Non-approved services are either not implemented, or the module itself explicitly prevents (via configuration defined in the Security Policy) an operator from using any non-approved service or service in a non-approved manner. In this case, an explicit indication via the use of a static code or an implicit indication via the successful completion of a service is sufficient.

3) For those modules implementing both approved and non-approved security services, a shared indicator common to multiple approved security services can be used. However, if the module provides support for running multiple security services simultaneously (that can alternate between approved & non-approved security service), then the module should clearly make a distinction between the running services and provide the indicator accordingly. For example, a module running in a multithreaded environment can use a dedicated status output as a service indicator where individual bits of the status output value correspond to individual running threads and denote their respective status. Another example is for the module that supports two different data input interfaces

(e.g. USB and Ethernet) that process two different service requests simultaneously. These can use separate status output interfaces to distinguish their respective indicators.

4) To serve as the approved security service indicator, the approved mode indicator under FIPS 140-2 **AS01.04** should be used together with other parameters, such as status output of the approved security service, to denote successful completion of the service. A mode indicator implemented for FIPS 140-2 Level 3 or 4 modules alone is not sufficient since this indicator will always be ON as soon as the module transitions to approved mode, irrespective of whether any service is performed or not.

The table below demonstrates for the example scenarios how the interpretation of the indicator provided by the module may be documented in the module's Security Policy. The return codes in the table are examples. In addition, security considerations such as default versus secure operation should be addressed.

Example scenarios	**Indicator provided by the module**	**Interpretation provided by the Security Policy**
1) Separate indicator per approved security service	Return code from an approved security service call	Return Code "0" denotes use of approved security service
	Logging a message in the file when an approved security service is used	Status message in the log file denotes use of approved security service
2) Global indicator for modules having approved services only	A static code	The static code "0" denotes use of an approved service
	A status code indicating the completion of service	The successful completion of a service is an implicit indicator for the use of an approved service
3) Shared indicator for multiple approved security services	Use of a dedicated bit from the module's status output value or a status register	Bit "X" set to "1" denotes use of approved service
	Use of a dedicated query function for the operator to determine whether the current security service in use is approved	Return value "1" from the query function denotes use of approved service
	Use of a dedicated status output interface such as LED	LED "X" turning green denotes use of approved service
4) Re-purpose approved mode indicator under FIPS 140-2 AS01.04 for approved security service indicator	Approved mode indicator (when implemented, required by Overall Security Ratings 3 and 4) AND service status output	Successful completion of a security service when the module is in approved mode denotes use of approved security service

Additional Comments

1. The module's current implementation of the approved security service indicator may imply different levels of complexity in addressing future algorithm transitions. Some may have more overhead than others. As a result of such transitions, the module will need to be updated to change the service indicator functionality to correctly reflect the new status of the disallowed algorithm. Modules following approaches listed in example scenarios 2), 3) and 4) are likely to require more overhead where the indicator mechanism is implemented at the module level, as compared to the modules

following the example scenario 1) that have the indicator implemented at the individual approved security service level.

2. It is the responsibility of the operator of the module to retrieve, recognize and take action on the service indicator. A FIPS 140-3 compliant module has the build-in service indicator capable of indicating the use of approved security services, but it may require the operator (e.g. human users, calling application, client process) to request for this information.
3. The service indicator **shall** be consistent with the approved implementation of security algorithms with FIPS 140-3 requirements met, including IGs (e.g. IG C.H), self-tests, etc. For instance, an approved algorithm (e.g. PBKDF is an approved KDF) **shall** not be considered as an approved implementation in the module if it does not have a CAVP certificate or does not include its required self-tests. Then the key derivation services using this PBKDF **shall** not be indicated as approved.
4. This additional comment applies to example scenarios 1) and 3) when the module:
    a. implements a non-approved mode,
    b. outputs a (non-required) indicator for the use of non-approved cryptographic services (e.g., a return code per non-approved service), and
    c. does not implement an approved mode indicator like the one described in example scenario 4) where the operator can unambiguously determine if the module is strictly operating in approved mode or in non-approved mode.

    If any non-approved cryptographic service utilizing a non-approved cryptographic algorithm, security function or process uses a (non-required) indicator identical to the (required) approved security service indicator, then TE02.24.02 isn't met: the (required) approved security service indicator does not permit to unambiguously differentiate the approved security services from the non-approved cryptographic services. The Security Policy cannot be used to resolve an ambiguous approved security service indicator when TE02.24.02 isn't satisfied by the module's implementation.
5. This IG makes it clear that an indicator is required for every approved security service as defined in Group 1, but not necessarily for services that fall under Groups 2-5. An API may be the approved security service (e.g., if a module is a cryptographic library), and in this case an indicator is required for each API call (i.e., if not otherwise unambiguously covered by a global indicator). But if a service uses underlying APIs, those APIs do not necessarily need an indicator, since the service that uses the APIs would indicate if the service is approved or not (e.g., as in the SHA-1 example in this IG).

Section 3 – Cryptographic module interfaces

3.4.A Trusted Channel

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	May 16, 2022
Relevant Assertions:	AS03.16, AS03.17, AS03.18 AS03.19, AS03.20, AS03.21, AS03.22, AS06.27, AS09.20. AS09.21, ASA.01, ASB.01
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

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 Overall Security Ratings 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 unauthorised 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 Overall Security Ratings 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:
    a. the physical characteristics of the Trusted Channel, with an explanation of how the Trusted Channel will protect the plaintext CSPs,
    b. 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,
    c. operator instructions for setup and operation of the Trusted Channel,
    d. the specific characteristics and specification of the source or target of the Trusted Channel relative to the cryptographic module.
    e. how the operator stays in control over the physical path and is able to prevent any unauthorised tampering.

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.

Section 4 – Roles, services, and authentication

4.1.A Authorised Roles

Applicable Levels:	2, 3, and 4
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	March 26, 2024
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 [for CSPs only], 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:

a. The hash algorithms when hashing data (i.e., not SSPs) that are specified in FIPS 180-4 and FIPS 202;

b. The deterministic random number generators and entropy sources that are specified in SP 800-90Arev1 and SP 800-90B;

c. Digital signature verification, as specified in "Digital Signature Standard", FIPS 186-2, FIPS 186-4, and FIPS 186-5;

d. Authentication procedures used for authenticating the operator and/or initialization procedures to setup the operator's authentication credentials; and

e. Show status, show version and self-tests.

This is to ensure the authorization is verified/granted using the module's authentication mechanism(s) for all non-exempt services; i.e., to be considered authorized, an operator must be authenticated.

Exceptions for other services that do not affect the security of the module may be claimed; however, in this case a justification, subject to CMVP approval, **shall** demonstrate the rationale in Additional Comment 1 below is met. It is recommended that requests for exception should be submitted to the CMVP via the existing Request for Guidance process detailed in the Management Manual Section 2.5 *Request for Guidance from CMVP*. Approval obtained prior to report submission can be referenced therein.

Additional Comments

1. The rationale for the stated exceptions is either:

a. that the referenced algorithms and services do not create, disclose, modify, substitute, access, or make use of the module's CSPs, and PSPs are not modified or substituted; or

b. that the referenced algorithms and services do not affect the security of the module, or the security of the information being protected by the module.

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. Please note the following rationale for the inclusion of the DRBG in the resolution exceptions above: 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 zeroisation 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.

6. For services that can update module code and only use cryptographic algorithms falling under an exception above (in c.), the operator is still required to be authenticated (at Security Level 2+). Loading and running new or additional code into the module, even if the code succeeds the required load test, has security implications. Services providing such capability are not granted an exception under this IG.

7. A module may establish a secure connection (e.g., TLS utilizing approved security functions) for protecting data exchanges with the operator before this operator is authenticated to an authorised role. This way, when authentication data is sent to the module, it benefits from the secure connection protection. Establishing a secure connection for the purpose of protecting authentication data and other data in-transit can be considered "authentication procedures used for authenticating the operator". As described in exception d. above, no operator authentication is required prior to executing such procedures. When a module establishes a secure connection without having authenticated the operator and claims Security Level 2, 3, or 4 for Section 7.4 (as allowed by this IG, exception d.), the module **shall** authenticate the operator with authentication data received under this same secure connection prior to executing any service that requires authentication.

8. When a module claims Security Level 2 for section 7.4, for an operator to be authenticated to one or more roles and perform services associated with that role or set of roles, this operator **shall** present the role-specific required authentication data for the assumption of the role or roles, as defined in the vendor documentation. An operator can only authenticate themselves to the module, i.e., their role or set of roles. Approved cryptographic services that require authentication cannot be associated with un-authenticated roles, unless they can claim an exception per this IG. A service requiring authentication may enable/disable other services that don't require authentication such that any unauthenticated operator (or operators assuming an unauthenticated role) may perform those services when they are enabled. However, when a service requiring authentication can unlock other services that must also require authentication under this IG (e.g., user data encrypt/decrypt service) but that don't implement operator authentication on their own (i.e., they can be executed by the same or any subsequent un-verified operator once enabled), then this is considered a **lock-based authentication model** where controlled or trusted operator(s) access is assumed outside the module while security services are unlocked. In this case, a submission claiming role-based (Level 2) authentication:

a. **shall** include the Caveat "No operator authentication is enforced for executing security services that were unlocked by an authenticated service" (see CMVP Caveats webpage),

b. **shall** confirm that power cycling the module disables or re-locks the previously unlocked security services, and

c. **shall** explain in Section 4 of the Security Policy the expected operator(s) access that justifies why this lock-based authentication model remains secure; for example,

    i. goal is to protect data-at-rest only,

    ii. the currently authenticated operator remains in physical control of the module interfaces until power off,

    iii. the host O.S. acts as the operator and is considered a trusted machine,

    iv. services are being unlocked as needed and re-locked immediately such that they remain unlocked only for a moment (in milliseconds) every time,

    v. all module interfaces were setup to interact only with trusted endpoints so when services are unlocked, they are strictly executed by those known endpoint operators,

    vi. etc.

4.4.A Multi-Operator Authentication

Applicable Levels:	*2, 3 and 4*
Original Publishing Date:	*September 21, 2020*
Effective Date:	*September 21, 2020*
Last Modified Date:	*September 21, 2020*
Relevant Assertions:	*AS04.57, AS04.58, AS04.59*
Relevant Test Requirements:	*TE(s) associated with AS(s) above*
Relevant Vendor Requirements:	*VE(s) associated with AS(s) above*

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 four 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.

5. 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 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 under Flags. 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	March 17, 2023
Relevant Assertions:	AS05.05 to AS05.23, AS10.17, AS10.18,
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

ISO/IEC 19790:2012 Section 7.10.2.2:

All software and firmware components within the cryptographic boundary **shall [10.17]** be verified using an approved integrity technique or EDC satisfying the requirements defined in [Section] 7.5. If the verification fails, the pre-operational 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 10 years.

The tester **shall** verify (in TE02.03.02 if **ISO/IEC 19790:2012** Section 5 is N/A, and if Section 5 is appliable, in TE05.05.01 or TE05.06.01) that vendor provided documentation confirms that the memory technology will not have any change or degradation of data for a minimum of 10 years after manufactured date.

The module's end of life procedures (**AS11.36** and **AS11.37**) **shall** be followed prior to the degradation of the memory technology as specified by the vendor provided documentation and the module's security policy **shall** state what these are and the timeline for these end of life procedures.

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

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

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 for Section 7.7 (*Physical security*) 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 for Section 7.7.

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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	July 26, 2024
Relevant Assertions:	AS09.08
Relevant Test Requirements:	TE09.08.01-2
Relevant Vendor Requirements:	VE09.08.01-2

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, and wherein the entropy data comes from outside of the TOEPP (Tested Operational Environment's Physical Perimeter, from IG 9.5.A), 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 via the entropy source's GetEntropy() interface.

   Examples include:

   (a) A hardware module with an entropy generating source 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 Security Policy **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 (e.g., keys) whose strengths are modified by available entropy.* The comparable cryptographic strength of an SSP is addressed under the Additional Comments below.

   (b) A software, firmware, or hybrid module that contains an approved DRBG, that is seeded exclusively from one or more known entropy sources, located within the TOEPP (from IG 9.5.A). For instance, a software library on a Linux platform making a call to a **SP 800-90B** entropy source within the module's TOEPP 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 Security Policy **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 (e.g., keys) whose strengths are modified by available entropy*.

   (c) A software, firmware, or hybrid module that contains an approved DRBG that issues a GET command to obtain the entropy from a source located outside the module's TOEPP.

   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 module **shall** enforce minimum length of entropy input requested and the Security Policy **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 (e.g., keys)*.

   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 from an entropy source **outside** the TOEPP of the module (or physical perimeter / cryptographic boundary for a hardware module), and while the module exercises no control over the amount or the quality of the obtained entropy. The module is assumed not to have

direct access to the entropy source's GetEntropy() interface. This scenario is expected to be a very rarely applicable (e.g., seed loaders, or preloaded seed).

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 that captures entropy data from outside the module's cryptographic boundary or the seed is pre-loaded within the module at factory.

What is required: (i) the Security Policy **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 there is knowledge about the entropy source used, the module **shall** enforce minimum length of entropy input accepted and guidance **shall** be provided in the Security Policy that the caller shall ensure that the entropy input provided has a minimum of 112 bits of entropy. (ii) The following caveat **shall** be added to the module's certificate: *No assurance of the minimum strength of generated SSPs (e.g., keys)*.

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, firmware, or hybrid module that contains an approved DRBG that receives a LOAD command (or its logical equivalent) with entropy obtained from an external entropy source (via an I/O port) that is outside the TOEPP of the module.

What is required: (i) the Security Policy **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 there is knowledge about the entropy source used, the module **shall** enforce the minimum length of entropy input accepted and guidance **shall** be provided in the Security Policy that the caller shall ensure that the entropy input provided has a minimum of 112 bits of entropy. (ii) the following caveat **shall** be added to the module's certificate: *No assurance of the minimum strength of generated SSPs (e.g., keys)*.

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 is passively receiving the entropy from an entropy source **inside** the TOEPP of the module. The module does not have direct access to the entropy source's GetEntropy() interface.

   Examples include:

   (a) An entropy source exists inside the TOEPP of the software, firmware, or hybrid module. A component inside the operational environment but outside of the cryptographic boundary of the module collects entropy data from the entropy source and stores that data in a storage space (e.g., a buffer, or a file later read by the module) that can be inside or outside of the cryptographic boundary of the module. The module contains an approved DRBG that receives entropy data from that storage space via a LOAD command (or its logical equivalent).

   This module **cannot** be validated. The entropy source does exist inside the TOEPP; thus the entropy source is well-defined. To obtain entropy data from an entropy source inside the TOEPP, the module **shall** have direct access to the entropy source's GetEntropy() interface. The module **shall not** rely on a component outside of the cryptographic boundary of the module, or on an operator of the module, to access the entropy source's GetEntropy() interface and obtain entropy data and deliver this entropy data to the module.

4. The module uses a *hybrid* approach to obtaining entropy for SSP generation. Some entropy is passively received from sources outside of the TOEPP of the module (or physical perimeter / cryptographic boundary for a hardware module), 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 TOEPP (or physical perimeter / cryptographic boundary for a hardware module). 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 workings of the entropy source within the operational environment and be able to verify the vendor's claim about the amount of entropy loaded into the 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 Security Policy **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 (e.g., keys).*

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 Security Policy **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 (e.g., keys)*.

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 (e.g., keys)*" 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 Security Policy **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 (e.g., keys)*" 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. At this time, the claimed security strength of any approved algorithm may not be greater than 256 bits. Therefore, if the module's DRBG provides at least 256 bits of security strength, the entropy caveats that end with "*whose strengths are modified by available entropy*" are not needed.

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

   The module's Security Policy **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 a single scenario in this IG. 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 (e.g., keys)* 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 (e.g., keys) whose strengths are modified by available entropy*?

   The approach taken in this IG is that when more than one caveat might be needed for a single scenario in this IG, 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 (e.g., keys)*. The scenarios of this IG are written following this single-caveat approach. The module's Security Policy **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.

7. The hybrid Scenario 4 above applies when two or more entropy sources are used together to seed a DRBG. If a module uses two or more entropy sources that independently seed the DRBG (i.e., the DRBG is seeded entirely from a single source per instantiation or re-seed), two or more of the scenarios described in the Resolution of this IG can apply separately. In this case, the module **shall** meet the requirements of each scenario independently and the certificate **shall** document the unique caveats that apply to each scenario. For example, one source may require the caveat "*No assurance of the minimum strength of generated SSPs (e.g., keys)*" while the other source requires the caveat "*The module generates SSPs (e.g., keys) whose strengths are modified by available entropy*", so both will be shown on the certificate. As another example, both entropy sources require the same caveat "*The module generates SSPs (e.g., keys) whose strengths are modified by available entropy*", so this caveat will only be shown once on the

certificate. The module's Security Policy **shall** inform the reader how many scenarios are applicable and include the appropriate documentation requirements for each scenario.

8. The following scenarios of this IG require an ESV and meet the applicable entropy requirements (i.e., IG D.J and IG D.K): 1(a), 1(b), 4 (for the source within the operational environment).
9. If the entropy source used by the module to claim available entropy is located within the TOEPP of the module (or physical perimeter / cryptographic boundary for a hardware module), ESV is required, and the module **shall** directly access the entropy source's GetEntropy() interface.
10. It is acceptable for the module to use callback strategies to obtain entropy data from the entropy source if the callback path still provides a direct connection between the module and the GetEntropy() interface. In other words, there needs to be a direct path between the GetEntropy() interface and the module if using the callback strategy, with no entity or code outside of the cryptographic boundary potentially manipulating the entropy data.
11. The entropy source used by the module may be a configurable item. E.g., the architecture-specific entropy source may be partially unspecified or non-configured until properly configured per the Security Policy (and ESV Public Use Document). Or a list of entropy sources and their ESV certificates may be provided in the Security Policy, and the final entropy source will be established per the configuration presented in the Security Policy and ESV Public Use Document. Yet, the entropy source(s) and its ESV certificate number(s) **shall** be present on the module certificate and in the Security Policy.
12. Until **January 1, 2026**, new module submissions to the CMVP can meet an earlier version of Resolution 2(b) of this IG, located at the following URL: https://csrc.nist.gov/CSRC/media/Projects/cryptographic-module-validation-program/documents/IG%209.3.A%20Resolution%202b%5BMarch%2026%202024%5D.pdf. This is a 'soft' transition in that modules that meet this earlier version of Resolution 2(b) will not be moved to the Historical list on the transition date.

Test Requirements

The vendor and tester evidence **shall** be provided under TE09.08.01 and TE09.08.02 and include the applicable scenario of this IG with justification.

9.5.A SSP Establishment and SSP Entry and Output

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	April 18, 2025
Relevant Assertions:	AS03.16 – AS03.22, AS09.10 – AS09.24
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

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 keys [1] 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 (note that the requirements for key versus non-key CSPs differ and are outlined in the footnotes below). 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**

[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.

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 / EE [3] – 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**
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**

[3] All AD / EE entries assume automated entry/output methods (without manual intervention or input) and therefore require an approved or allowed method listed in IG D.G and IG D.F. All other configurations assume manual entry (requiring human operator manipulation).

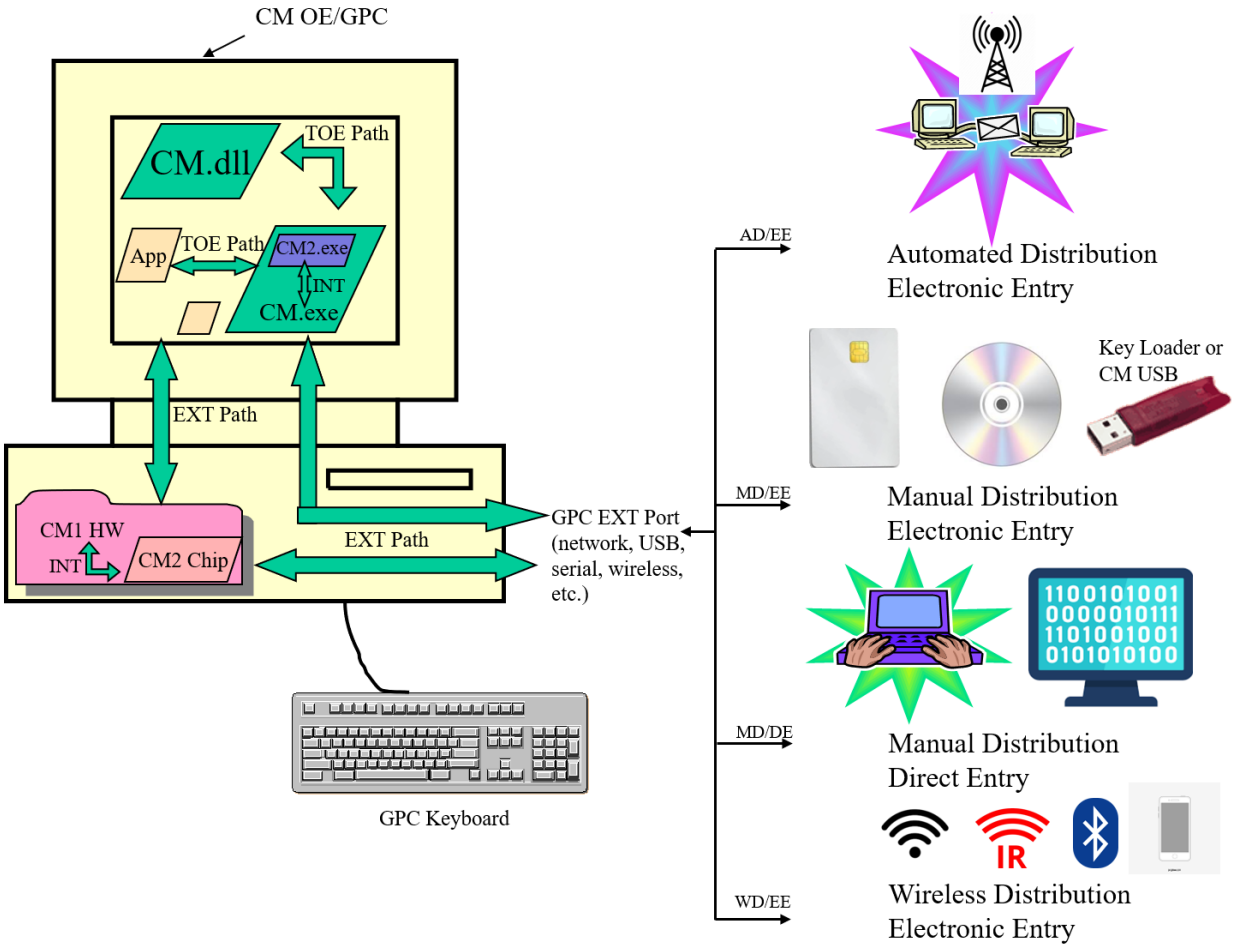
WD / EE – CM to/from CM via local Wireless [4] EXT Path	
CM Software[2] to/from CM Hardware or CM Software[2] via local Wireless EXT Path	**WD / EE**
CM Hardware to/from CM Hardware via local Wireless EXT Path	**WD / EE**
N/A [5] – CM to/from INT Path	
CM Software[2] to/from CM Software[2] via INT Path (CM embedded within CM)	**N/A**
CM Hybrid Software[2] 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 [6]**
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.

[6] The N/A means SSP establishment methods are not required (i.e., ISO-19790 section 7.9.4-5 requirements are optional) in the case where SSPs cross the cryptographic boundary of the sub-chip subsystem without crossing the SoC/single chip physical perimeter. However, if SSP establishment methods are implemented and their related algorithms are claimed in the submission, then related assertions (e.g., AS09.10) and standard requirements (e.g., from corresponding NIST Special Publication) are applicable.

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 [7]	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:

P [8]: Plaintext

E [9]: 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

TC [10]: Trusted Channel per **ISO/IEC 19790:2012** Section 7.3.4 and IG 3.4.A

SK: Split Knowledge using either E or a TC for the key

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

Additional Comments

---

[7] 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."

[8] 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."

[9] 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.

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

This IG reaffirms that SSPs established using *manual* distribution 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 for Section 7.9.

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

AS06.08: (Leve1 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	April 18, 2025
Relevant Assertions:	AS09.28
Relevant Test Requirements:	TE's associated with AS above
Relevant Vendor Requirements:	VE's associated with AS above

Background

Rules for SSP storage are described in general terms in **ISO/IEC 19790:2012**. 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 zeroisation 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 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 encryption using any approved mode of AES 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 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.
3. 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.
4. If the AES encryption is used to protect a stored SSP, the key encryption key may be established as shown in **SP 800-132**.
5. The **SP 800-131A** notation in this Implementation Guidance refers to the latest published revision of this standard.
6. 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 Zeroisation of One Time Programmable (OTP) Memory

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

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 SSPs.

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 zeroisation.

9.7.B Indicator of Zeroisation

Applicable Levels:	Level 2, 3, 4
Original Publishing Date:	May 4, 2021
Effective Date:	May 4, 2021
Last Modified Date:	May 4, 2021
Relevant Assertions:	AS09.33
Relevant Test Requirements:	TE09.33.01, TE09.33.02
Relevant Vendor Requirements:	VE09.33.01

Background

ISO/IEC 19790:2012

Terms and definitions

3.122 status information
information that is output from a cryptographic module for the purposes of indicating certain operational characteristics or states of the module.

7.3.3 Definition of interfaces
Status output interface. All output signals, indicators (e.g. error indicator), and status data (including return codes and physical indicators such as visual (display, indicator lamps), audio (buzzer, tone, ring), and mechanical (vibration)) used to indicate the status of a cryptographic module **shall [03.11]** exit via the "status output" interface. Status output may be either implicit or explicit.

7.4.3.1 Services general requirements

A cryptographic module **shall [04.12]** provide the following services to operators.

e) *Perform zeroisation.* The cryptographic module **shall [04.17]** perform zeroisation of the parameters as specified in 7.9.7.

7.9.7 Sensitive security parameter zeroisation
SECURITY LEVELS 2 AND 3 [11]
A module **shall [09.28]** provide methods to zeroise all unprotected SSPs and key components within the module.
Temporary SSPs **shall [09.32]** be zeroised when they are no longer needed.
The module **shall [09.33]** provide an output status indication when the zeroisation is complete.

7.6.3 Operating system requirements for modifiable operational environments

SECURITY LEVEL 2

- the cryptographic module **shall [06.26]** provide the following events to be recorded by the audit mechanism of the operating system:
  - modifications, accesses, deletions, and additions of cryptographic data and SSPs;
  - attempts to provide invalid input for Crypto Officer functions;
  - addition or deletion of an operator to and from a Crypto Officer role (if those roles are managed by the cryptographic module);
  - the use of a security-relevant Crypto Officer function;
  - requests to access authentication data associated with the cryptographic module;

[11] Security Level 4 incorporates this requirement without adding any additional requirements for zeroisation status

- the use of an authentication mechanism (e.g. login) associated with the cryptographic module; and
- explicit requests to assume a Crypto Officer role.

ISO/IEC 24759:2017

AS04.17: (Services — Levels 1, 2, 3, and 4)
The cryptographic module shall perform zeroisation of the parameters as specified in *{ISO/IEC 19790:2012}* 7.9.7.
NOTE This assertion is not separately tested.

AS09.33: (Sensitive security parameter zeroisation — Levels 2, 3, and 4)
The module shall provide an output status indication when the zeroisation is complete.

VE09.33.01: The vendor documentation shall specify that the module provides an output status indication when the zeroisation is complete *{AS03.11}*

TE09.33.01: The tester shall verify that the vendor provides documentation that specifies that the module provides an output status indication when the zeroisation is complete.

TE09.33.02: The tester shall perform zeroisation and verify the status output indicator.

AS04.35: (Software/Firmware loading — Levels 1, 2, 3, and 4)
All SSPs shall be zeroised prior to execution of the new image.

VE04.35.01: If a complete image replacement is supported, the vendor provided documentation shall specify that SSP zeroisation takes place prior to execution of the new image.

VE04.35.02: If a complete image replacement is supported, the vendor documentation shall specify the following SSPs zeroisation information:
a) zeroisation techniques;
b) rationale explaining how the zeroisation technique is performed in a time that is not sufficient to compromise SSPs.

AS06.26: (Operational environment — Level 2)
The cryptographic module shall provide the following events to be recorded by the audit mechanism of the operating system:

— modifications, accesses, deletions, and additions of cryptographic data and SSPs;
— attempts to provide invalid input for Crypto Officer functions;
— addition or deletion of an operator to and from a Crypto Officer role (if those roles are managed by the cryptographic module);
— the use of a security-relevant Crypto Officer function;
— requests to access authentication data associated with the cryptographic module;
— the use of an authentication mechanism (e.g. login) associated with the cryptographic module;

— explicit requests to assume a Crypto Officer role.

Question/Problem

Modules are required to zeroise an SSP for a variety of reasons. **ISO/IEC 19790:2012** requires that modules **shall** zeroise unprotected SSPs in the following scenarios:

- Automatic tamper response (levels 3 and/or 4). [**AS07.23, AS07.30, AS07.57, AS07.70**]
- Environmental failure protection (levels 3 & 4). [**AS07.81**]
- Maintenance; when entering or exiting maintenance mode (all levels). [**AS04.07**] and when performing physical maintenance (all levels). [**AS07.16**]
- As a result of the zeroisation service (all levels). [**AS04.17**]

In addition, the zeroisation of protected and unprotected SSPs **shall** be performed in the following scenarios:

- When loading a new software or firmware as a complete image replacement. [**AS04.35**]
- As a result of the level 4 zeroisation service. [**AS09.37**]
- For temporary SSPs when they are no longer needed (levels 2, 3 & 4). [**AS09.32**]
- For temporary value(s) generated during the integrity test of the module's software or firmware upon completion of the integrity test. [**AS05.10**]

Which purposes require the status indication from **AS09.33** and what may that indication be?

Resolution

1. Zeroisation of temporary values from the integrity test is a routine module operation and does not require a status output indication of zeroisation per **AS05.10**.
2. At level 1, zeroisation of unprotected SSPs may be performed procedurally by the module operator, and independent of the module's control (e.g., reformatting of a hard drive, the atmospheric destruction of a module during re-entry, etc.). The vendor's zeroisation procedure **shall** state how the operator will determine that zeroisation has completed. For level 1, the successful completion of the procedural zeroisation suffices as the implicit indicator that zeroisation has completed. The Security Policy **shall** document these procedures to zeroise unprotected SSPs and how the operator will determine whether the procedures were successful.
3. At levels 2 – 4, temporary SSPs **shall** be zeroised when they are no longer needed. This zeroisation **shall** have either an implicit or explicit status output indicating the zeroisation has completed. An implicit indicator, for instance, is an indicator from the status output interface that the function or service responsible for creating and zeroising the temporary SSPs has completed.
4. All other zeroisation performed by the module (e.g., for physical purposes, as tamper response, environmental failure protection, maintenance, or as a result of the zeroisation service) **shall** have either an implicit or explicit status output indicating the zeroisation has completed.
5. The zeroisation indicator may be a physical indicator (e.g., an LED) or a logical one (e.g., a return value, displayed message, etc.) on the status output interface. The zeroisation indicator may be a single unified indicator for the zeroisation of all affected SSPs or may be multiple indicators for groups of, or even individual, SSPs. The zeroisation indicator may be implicit as the normal, non-error, status output of the function performing zeroisation.
6. Modules that meet Security Level 2 for Operational Environment must record deletion/zeroisation of SSPs to the operating system's audit mechanism [**AS06.26**] and this may also be considered their zeroisation indicator. This resolution is not applicable to deletion/zeroisation of temporary SSPs.
7. The vendor documentation **shall** specify that the module provides an output status indication when the zeroisation is complete, and **shall** document all the module's implicit or explicit zeroisation status indicators.
8. The "Sensitive security parameters management" section of the Security Policy **shall** indicate and provide details on whether a SSP is zeroised implicitly or explicitly.

Additional Comments

1. Different methods of zeroisation within a module, or zeroisation for different purposes, may all use the same indicator of zeroisation or may have different zeroisation indicators.
2. Aside from modules that meet Security Level 2 for Operational Environment, zeroisation indicators are not required to be persistent or logged.
3. Return codes, or other indicators that are not output from the cryptographic boundary via the defined status output interface do not meet the requirement for a zeroisation indicator.
4. Per **ISO/IEC 19790:2012** section 7.9.7, parameters used solely for self-test purposes in 7.10 need not meet zeroisation requirements, and so zeroisation indicators would not apply.

Section 10 – Self-tests

10.2.A Pre-operational Integrity Technique Self-test

Applicable Levels:	All
Original Publishing Date:	March 17, 2023
Effective Date:	March 17, 2023
Last Modified Date:	March 17, 2023
Relevant Assertions:	AS10.20
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

ISO/IEC 19790:2012 Section 7.10.1:

A cryptographic algorithm that is used to perform the approved integrity technique for the pre-operational software/firmware test **shall [10.20]** first pass the cryptographic algorithm self-test specified in 7.10.3.2.

AS10.20 applies when an approved integrity technique is implemented. An approved integrity technique is not required for:

1. Hardware modules that contain no software or firmware in which case no integrity test is required.
2. Software and firmware components of a hardware module since an EDC is sufficient per **AS05.06.**
3. Software or firmware components within a disjoint hardware component of a hybrid module since an EDC is sufficient per **AS05.06**.

AS10.20 does not apply in cases where an approved integrity technique is implemented in non-reconfigurable memory, provided it meets IG 5.A.

Question/Problem

What are some examples to meet the self-test requirement of **AS10.20**?

Resolution

1. The most obvious way to meet **AS10.20** is for a module to have a dedicated algorithm self-test that is separate from the integrity test that meets the CAST requirements of IG 10.3.A. This may not necessarily need to strictly match the cryptographic algorithm used for the integrity test if permitted by IG 10.3.A. For example, a module utilizes RSA 2048 SigVer with SHA-224 for the integrity test but executes an RSA 3072 SigVer with SHA-256 CAST beforehand. IG 10.3.A is met since it requires that at a minimum, one approved RSA modulus size and hash size is tested, both sizes being used by the module (e.g., in an approved service), and the SHA-256 self-test covers the SHA-224 self-test requirement. The exact cryptographic algorithm used by the software/firmware integrity test (i.e., RSA 2048 SigVer with SHA-224) isn't being tested here, but **AS10.20** is still met.
2. Another solution is for a module to run the integrity test twice to meet the requirements of **AS10.20**, with the first integrity test counting as the CAST and the second as the integrity test. The first test can be further simplified by running on a smaller set of code data (n.b. any keyed algorithm **shall** be tested with at least twice as much data, in bits, as the strength of the key) and have an intermediate expected value. If the comparison is equal, it passes, and the module can stream the code through the algorithm for the full module's integrity test required by Section 5 of **ISO/IEC 19790:2012**.

Additional Comments

1. The provisions of this IG only apply to **AS10.20**. The CAST requirements are specified in IG 10.3.A.
2. The periodic self-test requirements specified in **ISO/IEC 19790:2012** 7.10.3.8 and further clarified in IG 10.3.E are applicable to this IG in that the pre-operational integrity technique self-test is considered a conditional CAST while the integrity test is a pre-operational self-test. For example, if the module performs periodic self-tests on the pre-operational self-tests (7.10.2 of **ISO/IEC 19790:2012**), then the cryptographic algorithm used for the approved integrity technique would not need to re-run its conditional CAST (7.10.3.2 of **ISO/IEC 19790:2012**).

10.3.A Cryptographic Algorithm Self-Test Requirements

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	April 18, 2025
Transition End Dates	**December 31, 2020 – for select algorithms**
Relevant Assertions:	AS10.01-06, AS10.25-35
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

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 [test] ~~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:

1. symmetric-key encryption and decryption algorithms (**FIPS 197**, **SP 800-38** series, **SP 800-67rev2**)?
2. hash algorithms that are not keyed (**FIPS 180-4**)?
3. SHA-3 permutation-based and extendable-output functions (**FIPS 202**) or SHA-3 derived functions (**SP 800-185**)?
4. message authentication algorithms that use a key and a hash algorithm (**FIPS 198-1**)?
5. stateful hash-based signature schemes (**SP 800-208**)?
6. deterministic random number generators (**SP 800-90A**)?
7. entropy sources used for random bit generation (**SP 800-90B**)?
8. key derivation functions, such as: KBKDF (**SP 800-108**), PBKDF (**SP 800-132)** and KDA (**SP 800-56C rev1** or **rev2**)?
9. algorithms that are tested as a CVL (IG 2.4.B)?
10. RSA, ECDSA, EdDSA, or DSA signature algorithms (**FIPS 186-4**, **FIPS 186-5**)?
11. key agreement schemes (**SP 800-56Arev3** or **SP 800-56Brev2**)?
12. asymmetric key transport schemes (**SP 800-56Brev2**)?
13. vendor affirmed algorithms (see Management Manual 7.2 *CMVP requirements pertaining to testing and approved algorithms*)?
14. post quantum Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) (**FIPS 203**)?
15. post quantum Module-Lattice-based Digital Signature Algorithm (ML-DSA) (**FIPS 204**)?
16. post quantum StateLess Hash-based Digital Signature Algorithm (SLH-DSA) (**FIPS 205**)?

Resolution

The requirements of this IG apply prior to the first operational use of the cryptographic algorithm. The **ISO/IEC 19790:2012** 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 follows similar requirements but makes some changes as specified below:

1. for symmetric-key algorithms, such as SKIPJACK, Triple-DES, or AES:

   a. 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;

b. If the module implements a decryption function, this function **shall** be separately self-tested. If a KAT is used, 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.

c. In addition, and applying only to the AES algorithm, 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, FF1). 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. These examples only highlight the forward and inverse cipher function requirements; additional requirements as specified in 1.a and 1.b (above), and 1.d (below) still apply (e.g., self-testing AES GCM Encrypt and Decrypt).

The reason the above two paragraphs only apply to AES and not Triple-DES is because every approved Triple-DES mode involves running both a forward and an inverse application of the underlying DES algorithm.

d. Furthermore, since symmetric modes that provide authentication/integrity (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 underlying algorithm implementation (i.e. AES and Triple-DES) that is contained in 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 CAST. If item (i) is self-tested, then this covers the requirements for items (ii) and (iii); if item (ii) is self-tested, it covers items (iii).

(i) A CAST for one of the following AES authenticated encryption modes is required if implemented: GCM, CCM, CMAC or GMAC. A CAST for Triple-DES CMAC is required if this mode is implemented;

(ii) 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 KW is required if no other Triple-DES authenticated encryption modes are implemented and self-tested;

(iii) A CAST for one of the non-authenticated encryption modes (ECB, CBC, OFB, CFB, CTR, or AES-FF1 or 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 (item c.) 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 (items a. and b.) 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 (item c.).

Note1: The Triple-DES and SKIPJACK algorithms are approved algorithms only for legacy decryption so only a CAST for decryption is required.

2. if the module implements a SHS function (**FIPS 180-4**), 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.

3. if the module implements SHA-3 permutation-based and/or extendable-output functions (see IG C.C and **FIPS 202**) or SHA-3 derived functions (**SP 800-185**):

- 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) or **SP 800-185** (cSHAKE, KMAC, TupleHash and ParallelHash), no matter how many of these functions the module may be designed to use.
- If a **FIPS 202** or **SP 800-185** function is used as part of a higher-level approved algorithm that is tested by the CAVP, then the module **shall** either perform a CAST for this higher-level algorithm that uses the **FIPS 202** or **SP 800-185** function, or perform a CAST for this higher-level algorithm using a different function (if this configuration is implemented in the module) and have the **FIPS 202** or **SP 800-185** function implementation self-tested separately (either as a stand-alone algorithm or as part of the another higher-level algorithm's CAST). No additional CASTs for any of the **FIPS 202** or **SP 800-185** functions implemented by this module are required.
- If the module contains several Keccak-p permutation implementations, a CAST **shall** be performed for more than one implementation of the **FIPS 202** or **SP 800-185** functions so that each permutation implementation is self-tested separately.
- If the module supports both **FIPS 202** and **SP 800-185** functions and they share the same Keccak-p permutation implementation, then one CAST on a function defined in either **FIPS 202** or **SP 800-185** is sufficient to meet the CAST requirement for **FIPS 202** functions.

Note2: The reason for requiring a CAST for only one of the **FIPS 202** or **SP 800-185** functions is that all of these functions rely on the same underlying Keccak-p permutation and do not add significant complexity to this underlying function. Note that this is different from the SHA-1 and SHA-2 CAST requirements where separate CASTs are needed for SHA-1, SHA-256 and SHA-512, if the module is designed to use these hash functions.

Note3: Higher-level algorithms, in this context, are the algorithms such as digital signatures and the key establishment, but not the **FIPS 202** or **SP 800-185** algorithms.

Note4: removed.

4. if the module implements a HMAC function (**FIPS 198-1**), a CAST for HMAC is required and **shall** be performed with the HMAC function using at least one of the implemented underlying SHA-1, SHA-2, or SHA-3 algorithms.

5. if the module implements stateful hash-based signature schemes (**SP 800-208**):

   SP 800-208 specifies two stateful hash-based signature schemes that can be used to generate a digital signature: the Leighton-Micali Signature (LMS) system and the eXtended Merkle Signature Scheme (XMSS), along with their multi-tree variants, the Hierarchical Signature System (HSS) and multi-tree XMSS ($XMSS^{MT}$).

   - LMS and HSS
     - If the module implements LMS signature generation then it **shall** also implement key generation and **shall** have CASTs for both LMS signature generation and key generation.
     - If the module implements HSS signature generation then it **shall** also implement key generation and **shall** have CASTs for both HSS signature generation and key generation.
     - If the module implements signature verification for LMS, it **shall** have a CAST for it.
     - If the module implements signature verification for HSS, it **shall** have a CAST for it.
     - Since HSS builds on LMS, the HSS CASTs would cover the related LMS CAST requirements in the bullets above.
   - XMSS and $XMSS^{MT}$
     - If the module implements XMSS signature generation then it **shall** also implement key generation and **shall** have CASTs for both XMSS signature generation and key generation.
     - If the module implements $XMSS^{MT}$ signature generation then it **shall** also implement key generation and **shall** have CASTs for both $XMSS^{MT}$ signature generation and key generation.
     - If the module implements signature verification for XMSS, it **shall** have a CAST for it.
     - If the module implements signature verification for $XMSS^{MT}$, it **shall** have a CAST for it.
     - Since $XMSS^{MT}$ builds on XMSS, the $XMSS^{MT}$ CASTs would cover the related XMSS CAST requirements in the bullets above.
   - for the implemented SHA-256, SHA-256/192, SHAKE256/256, or SHAKE256/192 hash functions used within the above algorithms, the following **shall** be the minimal requirements:
     - a CAST for SHA-256 is required;
     - a CAST for SHA-256/192 is required if SHA-256/192 is implemented without SHA-256;
     - a CAST for SHAKE256/256 is required;
     - a CAST for SHAKE256/192 is required if SHAKE256/192 is implemented without SHAKE256/256;

     These hash CASTs can be satisfied as part of higher-level CASTs (e.g., LMS, HSS, XMSS, or $XMSS^{MT}$)

   Note5: For these schemes the complexity of testing is dependent upon the height of the tree(s) supported. If the module supports multiple parameter sets it is sufficient that the CASTs exercise the smallest parameter set (lowest height trees) supported by the module to provide assurance that the algorithm is operating correctly. In the case that the module supports only one parameter set in the higher order a CMVP RFG can be submitted for assistance with this or other unique cases.

Note6: Please refer to **SP 800-208** for requirements on the parameter sets for LMS and XMSS.

Note7: If the module is part of a distributed signature scheme per Section 7 of **SP 800-208**, the CASTs described above only apply to the functionality implemented within the module's boundary. For example, if multiple cryptographic modules implement the distributed multi-tree scheme for HSS, each module will need to individually implement and self-test only the LMS. Similarly, if multiple cryptographic modules implement the distributed multi-tree scheme for XMSS^{MT}, each module will need to individually implement and self-test the slightly modified versions of the XMSS key and signature generation algorithms (see **SP 800-208** Section 7.2.1).

Note8: Modules claiming compliance with **SP 800-208** and use of the approved security functions described therein **shall** meet, and make clear in the documentation, the conformance requirements noted in Section 8 of the standard including, but not limited to, the following that apply when a module implements key generation and signature generation:

*Cryptographic modules that implement signature generation for a parameter set **shall** also implement key generation for that parameter set. Implementations of the key generation and signature algorithms in this document **shall** only be validated for use within hardware cryptographic modules. The cryptographic modules **shall** be validated to provide FIPS 140-2 or FIPS 140-3 [19] Level 3 or higher physical security, and the operational environment **shall** be non-modifiable or limited. In addition, a cryptographic module that implements the key generation or signature algorithms **shall** only operate in an approved mode of operation and **shall not** implement a bypass mode. The cryptographic module **shall not** allow for the export of private keying material. The entropy source for any approved random bit generator [6] used in the implementation **shall** be located inside the cryptographic module's physical boundary.*

Please additionally refer to IG C.N *Requirements for SP 800-208 Schemes*.

6. 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 CASTs 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 KAT 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.

7. if the module implements an approved ENT or ESV (**SP 800-90B**), then the health-tests are required and **shall** be performed as specified in **SP 800-90B** Section 4.2, including the start-up health tests, the on-demand tests, and the continuous tests. These self-tests are considered CASTs (i.e., fault detection per AS10.34). See IG D.J Additional Comment 3 for specific requirements on vetted conditioning components used within a **SP 800-90B**-compliant entropy source.

8. Key derivation functions:

   - 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 (the key derivation functions and the authenticated encryption/decryptions, as applicable) implemented in the module that are used in the derivation or the protection of the Data Protection Key (DPK), if the same implementations of the underlying algorithms are not already self-tested either on their own or as part of higher level algorithm CASTs.

In addition, the lengths and the properties of the Password and Salt parameters, as well as the desired length of the Master Key used in a CAST **shall** be among those supported by the module in the approved mode. The Iteration Count parameter does not need to be among those supported by the module in the approved mode but **shall** be at least two. The reason is the Iteration Count is only used to repeat the same pseudorandom function calculation, and a large Iteration Count can greatly affect the performance of the CAST. There is enough assurance that the Iteration Count will perform correctly for any number if it runs successfully for at least two iterations.

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

- if the module implements an approved KDA (**SP 800-56C rev1** or **rev2**), 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 KDA CASTs do not require separate CASTs.

  In addition, the module **shall** self-test all underlying prerequisite algorithms used in the remaining **SP 800-56C rev1** or **rev2** 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 CASTs.

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

9. 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, this Implementation Guidance document may specifically require self-testing CVLs (such as in IGs D.F and D.G).

10. for each **FIPS 186-4** and **FIPS 186-5** public key digital signature algorithm (RSA, DSA, ECDSA, deterministic ECDSA signature generation, EdDSA, and HashEdDSA), 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 RSASSA-PSS-compliant scheme, it is not necessary to perform any additional CAST for the implementation of the digital signature compliant with RSASSA-PKCS1-v1.5, even if this scheme is also supported by the module. More information on CAST requirements for these algorithms are in the following bullets and notes.

    - for deterministic ECDSA signature generation,
        - the module **shall** have a deterministic ECDSA digital signature generation CAST.

    Note9: The signature verification CAST for signatures generated using deterministic ECDSA is met with an ECDSA signature verification CAST (specified below).

    Note10: the ECDSA signature generation and deterministic ECDSA signature generation methods **shall** each have their own CAST if both are implemented in the approved mode.

    - for the EdDSA algorithm,
        - if the module implements digital signature generation, the module **shall** have an EdDSA digital signature generation CAST. If a KAT is used, the EdDSA digital signature **shall** be pre-computed, generate an EdDSA digital signature using known data and keys, and then compare the result to the pre-computed value;
        - if the module implements digital signature verification, the module **shall** have an EdDSA digital signature verification CAST. If a KAT is used, the EdDSA digital signature **shall** be pre-

computed (which could be the output of the EdDSA digital signature generate test), and using a known key, verify the signature as specified in section 7.7 of **FIPS 186-5**.

- for the HashEdDSA algorithm,
  - if the module implements digital signature generation, the module **shall** have a HashEdDSA digital signature generation CAST. If a KAT is used, the HashEdDSA digital signature **shall** be pre-computed, generate a HashEdDSA digital signature using known data and keys, and then compare the result to the pre-computed value;
  - if the module implements digital signature verification, the module **shall** have a HashEdDSA digital signature verification CAST. If a KAT is used, the HashEdDSA digital signature **shall** be pre-computed (which could be the output of the HashEdDSA digital signature generate test), and using a known key, verify the signature as specified in section 7.8.2 of **FIPS 186-5**.

Note11: If both EdDSA and HashEdDSA are implemented, a CAST on either one of them is sufficient to meeting the CAST requirements for both. The reason is that both EdDSA and HashEdDSA support the same underlying cryptography, with the HashEdDSA using the same hash algorithm one more time than the EdDSA. Self-testing both digital signature generation and verification is required if both are implemented.

Note12: If both curves Ed25519 and Ed448 are supported (i.e., within EdDSA and/or HashEdDSA), CAST(s) **shall** cover both curves.

- for the RSA algorithm,
  - if the module implements digital signature generation, the module **shall** have an RSA digital signature CAST. If a KAT is used, 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;
  - if the module implements digital signature verification, the module **shall** have an RSA digital signature CAST. 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 hash with its target value.
  - 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 800-56Brev2** and IG D.G, the module is required to perform the **SP 800-56Brev2** CASTs as described in IG D.G.

  Note13: an RSA CAST **shall** be performed using both the public and private exponents (e and d) and the two exponents **shall** correspond [that is, $d * e = 1 \pmod{\text{LCM}(p\text{-}1, q\text{-}1)}$]. The public exponent e used in this RSA CAST **shall** be chosen from the public exponent values supported by the module.

  Note14: 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.

  Note15: 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 and verification (as supported by the module). For a KAT, this requires the randomization parameter be fixed.

  Note16: a CAST **shall** be performed at a minimum on any one approved modulus or curve size that is supported by the module.

Note17: an ECDSA CAST **shall** be performed at a minimum, on any one of the implemented NIST-recommended curves (and for which the CAVP certificate includes the curve) in each of the implemented two types of fields (i.e., prime field where *GF(p)*, and binary field where $GF(2^m)$).

11. key agreement schemes (see IG D.F, **SP 800-56Arev3** and **SP 800-56Brev2**):

    - CAST requirements for approved key agreement and shared secret computation schemes are shown in IG D.F.
    - all new module submissions after **December 31, 2020** **shall** comply with these CAST requirements.

    Note18: for the SSH KDF (CVL), self-testing the underlying SHA hash functions is sufficient to meet the SSH KDF CAST requirement. This allowance applies to the SSH KDF only and not the other protocol specific KDFs used within an approved key agreement scheme.

    Note19: for the IKEv1 KDF (CVL), the module **shall** self-test any one (1) of the three (3) modes specified in 4.1.1 of **SP 800-135rev1** which inherently also tests the underlying HMAC PRF. This allowance applies to the IKEv1 KDF only and not the other protocol specific KDFs used within an approved key agreement scheme.

    Note20: For the TLS KDFs (CVL), at minimum each of the following TLS KDF modes **shall** have their own CASTs (if implemented): TLS 1.0/1.1 KDF, TLS 1.2 KDF (RFC 5246/RFC 7627), and TLS 1.3 KDF. Note, per the requirement near the end of the Resolution section of this IG, if any of the KDFs are separately implemented (e.g., TLS 1.0 KDF and TLS 1.1 KDF, despite using the same underlying cryptography), they must have their own CASTs.

12. key transport schemes (see IG D.G and **SP 800-56Brev2**):

    - CAST requirements for approved key transport schemes are shown in IG D.G.
    - all new module submissions after **December 31, 2020** **shall** comply with these CAST requirements.

13. If an algorithm's implementation in the module is vendor-affirmed, then the algorithm **shall** be self-tested, unless an IG specifically removes or reduces the requirement.

14. Post quantum Module-Lattice-based Key-Encapsulation Mechanism (ML-KEM) (**FIPS 203**):

    - if the module implements ML-KEM encapsulation, the module **shall** have a CAST for the ML-KEM encapsulation mechanism. The encapsulation algorithm of ML-KEM accepts an encapsulation key (*ek*) as input, requires randomness, and outputs a ciphertext (*c*) and a shared secret (*K*). The CAST **shall** use the ML-KEM encapsulation algorithm (i.e., Algorithm 20 in **FIPS 203**) or ML-KEM encapsulation internal algorithm (i.e., Algorithm 17 in **FIPS 203**), and for a KAT, using fixed/predetermined values (i.e., *ek* and *m*) to compare the resulting outputs to pre-computed values (i.e., *c* and *K*).
    - if the module implements ML-KEM decapsulation, the module **shall** have a CAST for the ML-KEM decapsulation mechanism. The decapsulation algorithm of ML-KEM accepts a decapsulation key (*dk*) and a ML-KEM ciphertext (*c*) as input, does not use any randomness, and outputs a shared secret (*K'*). The CAST **shall** use the ML-KEM decapsulation algorithm (i.e., Algorithm 21 in **FIPS 203**) or ML-KEM decapsulation internal algorithm (i.e., Algorithm 18 in **FIPS 203**), and for a KAT, using fixed/predetermined values (i.e., *dk* and *c*) to compare the resulting output to pre-computed value (i.e., *K'*).

      Note21: The ML-KEM decapsulation CASTs **shall** cover *both* the implicit rejection and (unnamed) non-rejection paths.

    Note22: The above CASTs **shall** be performed on at least one of the following parameter-sets for ML-KEM that are implemented in the approved mode: ML-KEM-512, ML-KEM-768, or ML-KEM-1024.

- if the module implements ML-KEM key generation, the module **shall** have an ML-KEM key generation CAST. The ML-KEM key generation does not take input and outputs an encapsulation key (*ek*) and a decapsulation key (*dk*). The CAST **shall** use the ML-KEM key generation algorithm (i.e., Algorithm 19 in **FIPS 203**) or ML-KEM key generation internal algorithm (i.e., Algorithm 16 in **FIPS 203**), and for a KAT, using a fixed/predetermined random values (i.e., *z* and *d*) to compare the resulting outputs to the pre-computed values (i.e., *ek* and *dk*).

- Additionally, the ML-KEM PCT requirements are clarified in Additional Comment 1 below.

15. Post quantum Module-Lattice-based Digital Signature Algorithm (ML-DSA) (**FIPS 204**):

- if the module implements digital signature generation, the module **shall** have an ML-DSA digital signature generation CAST for either the "hedged" *or* "deterministic" algorithm (per **FIPS 204** Section 5.2). ML-DSA signature generation algorithm takes as input a private key (*sk*), a message (*M*), and a context string (*ctx*), and it outputs a signature (σ). The CAST **shall** use the ML-DSA signature generation algorithm (i.e., Algorithm 2 in **FIPS 204**) or ML-DSA signature generation internal algorithm (i.e., Algorithm 7 in **FIPS 204** or a modified version of it per the PQC FAQ), and for a KAT, using fixed/predetermined values (i.e., for Algorithm 2: *sk*, *M*, *rnd*, and *ctx* which may be an empty string; for Algorithm 7: *sk*, *M'*, and *rnd*), to compare the resulting outputs to the pre-computed value (σ). If the module supports a modified version of Algorithm 7 (per PQC FAQ) *mu* may be fixed rather than *M'*.

  Note23: The ML-DSA digital signature generation CASTs **should** cover all applicable *rejection sampling loop* paths. For ML-DSA-65 and ML-DSA-87 this includes the checks during signature generation on *z*, r_0, and the number of 1s in *h*. For ML-DSA-44, this additionally includes the check on ct_0. A list of test cases based on the pseudocode in **FIPS 204** is provided at the following link to assist implementations in applying this recommendation: https://pages.nist.gov/ACVP/draft-celi-acvp-ml-dsa.html#name-ml-dsa-siggen-test-types. Implementations that deviate from the pseudocode may not be able to fully utilize these test cases. It is recommended to use the values from the list of test cases provided unless more applicable test cases are found by the vendor or CSTL.

  Note24: If both ML-DSA signature generation and pre-hash ML-DSA signature generation (Section 5.4 and Algorithm 4 in **FIPS 204**) are implemented and they both share the same underlying cryptographic implementations, a CAST on either one of them is sufficient to meet the CAST requirements for both.

- if the module implements digital signature verification, the module **shall** have an ML-DSA digital signature verification CAST. The ML-DSA signature verification algorithm takes as input a public key (*pk*), a message (*M*), a signature (σ), and a context string (*ctx*), and outputs a Boolean value (i.e., a value that is true if the signature is valid with respect to the message and public key, and false if the signature is invalid). The CAST **shall** use the ML-DSA signature verification algorithm (i.e., Algorithm 3 in **FIPS 204**) or ML-DSA signature verification internal algorithm (i.e., Algorithm 8 in **FIPS 204** or a modified version of it per the PQC FAQ), and for a KAT, using a fixed/predetermined values (i.e., for Algorithm 3: *pk*, *M*, σ, and *ctx* which may be an empty string; for Algorithm 8: *pk*, *M'*, and σ) to verify the signature (i.e., returns true). If the module supports a modified version of Algorithm 8 (per PQC FAQ) *mu* may be fixed rather than *M'*.

  Note25: If both ML-DSA signature verification and pre-hash ML-DSA signature verification (Section 5.4 and Algorithm 5 in **FIPS 204**) are implemented and they both share the same underlying cryptographic implementations, a CAST on either one of them is sufficient to meet the CAST requirements for both.

Note26: The above CASTs **shall** be performed on at least one of the following parameter-sets for ML-DSA that are implemented in the approved mode: ML-DSA-44, ML-DSA-65, ML-DSA-87.

- if the module implements ML-DSA key generation, the module **shall** have an ML-DSA key generation CAST. The ML-DSA key generation takes no inputs and outputs a public key (*pk*) and private key (*sk*). The CAST **shall** use the ML-DSA key generation algorithm (i.e., Algorithm 1 in **FIPS 204**) or ML-DSA key generation internal algorithm (i.e., Algorithm 6 in **FIPS 204**), and for a KAT, using a fixed/predetermined random seed (ξ) value, to compare the resulting outputs to the pre-computed values (i.e., *pk* and *sk*).

- Additionally, the ML-DSA PCT requirements are clarified by Additional Comment 1 below.

16. Post quantum StateLess Hash-based Digital Signature Algorithm (SLH-DSA) (**FIPS 205**):

- if the module implements digital signature generation, the module **shall** have an SLH-DSA digital signature generation CAST. SLH-DSA signature generation algorithm takes as input a private key (SK), a message (*M*), and a context string (*ctx*), and it outputs a signature (SIG). The CAST **shall** use the SLH-DSA signature generation algorithm (i.e., Algorithm 22 in **FIPS 205**) or SLH-DSA signature generation internal algorithm (i.e., Algorithm 19 in **FIPS 205**), and for a KAT, using fixed/predetermined values (i.e., for Algorithm 22: SK, *M*, *addrnd* if non-deterministic variant, and *ctx* which may be an empty string; for Algorithm 19: SK, *M'* and *addrnd* if non-deterministic variant) to compare the resulting outputs to the pre-computed value (i.e., SIG).

  Note27: If both SLH-DSA signature generation and pre-hash SLH-DSA signature generation (Section 10.2.2 and Algorithm 23 in **FIPS 205**) are implemented and they both share the same underlying cryptographic implementations, a CAST on either one of them is sufficient to meet the CAST requirements for both.

- if the module implements digital signature verification, the module **shall** have an SLH-DSA digital signature verification CAST. The SLH-DSA signature verification algorithm takes as input a public key (PK), a message (*M*), a signature (SIG), and a context string (*ctx*), and outputs a Boolean value (i.e., a value that is true if the signature is valid with respect to the message and public key, and false if the signature is invalid). The CAST **shall** use the SLH-DSA signature verification algorithm (i.e., Algorithm 24 in **FIPS 205**) or SLH-DSA signature verification internal algorithm (i.e., Algorithm 20 in **FIPS 205**), and for a KAT, using fixed/predetermined values (i.e., for Algorithm 24: PK, *M*, SIG, and *ctx* which may be an empty string; for Algorithm 20: PK, *M'*, SIG) to verify the signature (i.e., returns true).

  Note28: If both SLH-DSA signature verification and pre-hash SLH-DSA signature verification (Section 10.3 and Algorithm 25 in **FIPS 205**) are implemented and they both share the same underlying cryptographic implementations, a CAST on either one of them is sufficient to meet the CAST requirements for both.

Note29: If a module supports both SHA2 and SHAKE based parameter-sets in the approved mode, then the above CASTs **shall** be on at least two of the following parameter-sets for SLH-DSA: one that uses SHA2 and one that uses SHAKE.

If a module only supports SHA2 or SHAKE based parameter-sets in the approved mode, then the above CASTs **shall** be on at least one of the following parameter-sets for SLH-DSA: one that uses SHA2 or one that uses SHAKE, whichever is supported in the approved mode.

See **FIPS 205** Table 2 for a full list of approved parameters associated with each of the following algorithms:

- SLH-DSA-SHA2-128s, SLH-DSA-SHAKE-128s,
- SLH-DSA-SHA2-128f, SLH-DSA-SHAKE-128f
- SLH-DSA-SHA2-192s, SLH-DSA-SHAKE-192s
- SLH-DSA-SHA2-192f, SLH-DSA-SHAKE-192f
- SLH-DSA-SHA2-256s, SLH-DSA-SHAKE-256s

- SLH-DSA-SHA2-256f, SLH-DSA-SHAKE-256f

It is recommended (but not required) that if the module implements both "s" and "f" algorithms, the module self-test at least one of each "s" and "f" algorithm.

Note30: Similar to **SP 800-208** algorithms, for **FIPS 205** algorithms the complexity of testing is dependent upon the height of the tree(s) supported. If the module supports multiple parameter sets it is sufficient that the CASTs exercise the smallest parameter set (lowest height trees) supported by the module to provide assurance that the algorithm is operating correctly. In the case that the module supports only one parameter set in the higher order a CMVP RFG can be submitted for assistance with this or other unique cases.

- if the module implements SLH-DSA key generation, the module **shall** have an SLH-DSA key generation CAST. The SLH-DSA key generation takes no inputs and outputs a public key *(PK.seed, PK.root*) and private key (*SK.seed, SK.prf, PK.seed, PK.root*). The CAST **shall** use the SLH-DSA key generation algorithm (i.e., Algorithm 21 in **FIPS 205**) or SLH-DSA key generation internal algorithm (i.e., Algorithm 18 in **FIPS 205**), and for a KAT, using fixed/predetermined random values (i.e., SK.seed, SK.prf, and PK.seed) to compare the resulting outputs to the pre-computed values (i.e., public key and private key).

- Additionally, the SLH-DSA PCT requirements are clarified by Additional Comment 1 below.

General CAST requirements:

Note31: If the module contains different implementations (e.g., separate libraries, or two instances on the same or different hardware) of a single algorithm in an approved mode, each implementation of this algorithm **shall** be self-tested separately.

Note32: All other approved algorithms that have been issued a CAVP certificate **shall** be self-tested unless an IG specifically reduces the requirement.

Note33: All underlying approved algorithms used within a higher-level algorithm **shall** be self-tested, either with their own CAST or as part of a CAST for the higher-level algorithm. CAST exemptions specified in this IG apply equally to algorithms that are self-tested with their own CAST or as part of a CAST for a higher algorithm.

E.g., if a module implemented two versions of KBKDF, one using HMAC, and another using KMAC, and the module self-tests the KBKDF using HMAC but not the KBKDF using KMAC, the module would still need to meet the underlying KMAC CAST requirements per the requirements of this IG (i.e., under the **SP 800-185** section).

Another example supporting this rule would be having a single CAST for an RSA-OAEP scheme shown in Section 7.2.2.1 of **SP 800-56Brev2**, using any approved hash function, call it hash1, playing the role of H in the notation of this section of **SP 800-56Brev2** and implemented by the module for this purpose, and another implemented approved hash function, call it hash2 (which may be different from hash1) employed by the mask generation function (MGF). If the module also supports an RSA-OAEP scheme implementation with a different choice of hash functions (say, hash3 as H and hash4 in the MGF) then it is not necessary to perform a separate CAST for this RSA-OAEP implementation as long as hash3 and hash4 are each self-tested by some other acceptable means, i.e., as a stand-alone hash function or as part of a different higher-level algorithm.

Note34: 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 CAST 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.

Note35: Unless otherwise stated, CASTs **shall** be performed on algorithms and parameters (e.g., key sizes, curves, modes) that are used by an approved service (other than for self-testing). For example, a module that only supports AES-GCM with a 192-bit key as an approved service in the approved mode will require a CAST that uses a 192-bit key. (Using a 128-bit or 256-bit key will not cover the CAST requirements as the CAST does not include a key size used by an approved service in the approved mode.).

Note36: Unless otherwise stated, all the CASTs as mentioned in the Resolution in this IG can be met by any one of the following: *known answer* test (most common), *comparison* test (see below) or *fault-detection* test (currently not well-defined).

However, per TE10.33.01, *comparison* tests require the continuous comparison of each output of each cryptographic operation, having performed it on at least two independent implementations of the comparison tested algorithm. The *comparison* test compares the output of the independent algorithm implementations prior to any use of the output by the module or output of it from the module; and if the comparison fails (the output differs) the module goes into an error state without using or outputting the data which failed the *comparison* test. This continuous comparison, when applied to the first operational use of the algorithm will satisfy the requirement of **AS10.27** to self-test prior to first operational use; the module need not process test-only data through the algorithm *comparison* test prior to first operational use. Also note, a *comparison* test requires the randomization parameter(s), as applicable per algorithm, to be shared across all compared implementations.

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). At minimum, the PCT that is required by the underlying algorithm standard (e.g., **SP 800-56Arev3** Section 5.6.2.1.4 option 'b' or **SP 800-56Brev2** Section 6.4.1.1 option '2.') **shall** be performed, which will satisfy the PCT requirement in either TE10.35.01 for key transport, TE10.35.02 for signatures, or TE10.35.03 for key agreement. For approved algorithms from **FIPS 186-5**, **SP 800-56Arev3** or **SP 800-56Brev2**, or **FIPS 204**, if at the time a PCT on a key pair is performed it is known whether the keys will be used in a key agreement scheme, digital signature algorithm or to perform a key transport, then the PCT **shall** be performed consistent with the intended use of the keys (i.e., TE10.35.01 for key transport, TE10.35.02 [1] for signatures, or TE10.35.03 [2] for key agreement), even if the underlying standard does not require a PCT [3]. If at the time when the PCT is performed the keys' intended usage is not known, then any of the three PCTs described in AS10.35 **shall** be performed on this key pair.

   For key pairs generated for use with approved algorithms in **SP 800-208** and **FIPS 205**, the PCT (described by the tester in TE10.35.02) may be limited to confirming the same key identifier (*I* in the

[1] Note that a RSA PCT which satisfies TE10.35.02 (for signatures) also satisfies TE10.35.01 (for key transport), even if the intended usage is known (i.e., for signatures or key transport). However, the reverse is not true in that the TE10.35.01 PCT cannot satisfy the TE10.35.02 PCT requirement (assuming the intended usage is known by the module), for similar reasons as the RSA CAST specified in IG D.G (see Additional Comment #3).

[2] This PCT option may be simpler than the one defined in the algorithm standard (e.g., a PCT on the prerequisite DSA/ECDSA algorithms should be enough to cover the DH / ECDH PCT; or a PCT that mimics the shared secret computation using the recent generated key pair and a known key pair with the same domain parameters and comparing the shared secret computation values calculated using either FFC DH or ECC CDH primitives).

[3] For example, SP 800-56Arev3 Section 5.6.2.1.4 option 'a)' does not require an additional PCT if "… the owner generates the key pair as specified in Section 5.6.1".

case of LMS, *SEED* in the case of XMSS and *PK.SEED* for SLH-DSA) is shared by the resulting public and private key following generation [4].

For key pairs generated for use with approved KEMs in **FIPS 203**, the PCT (described by the tester in TE10.35.01) **shall** consist of applying the encapsulation key *ek* to encapsulate a shared secret *K* leading to ciphertext *c*, and then applying decapsulation key *dk* to retrieve the same shared secret *K*. The PCT passes if the two shared secret *K* values are equal.

The PCT **shall** be performed either when keys are generated/imported, prior to the first exportation, or prior to the first operational use (if not exported before the first use).

[4] For hash-based signatures (HBS), there are no weak keys or keys that could leak information about the key pairs if incorrectly generated. Unlike CAST which have the ability to choose the key sizes used to minimize execution time, PCT are required to use the key being generated. As such, the execution time for a sign followed by verify operation is directly linked to the parameters of the key being generated. In the case where a higher complexity key (e.g. high number of tiers and/or high number of levels) is generated, performing a traditional sign followed by verify operation as a PCT could take significant time to execute (there is a real possibility this could take hours to days for large keys). The selected Winternitz chain length which is set at key generation time, also impacts signature and verification time.

The number of possible signatures with a given HBS key, the time to perform a signature, and the size of the signature are trade-offs when selecting target parameter sets for HBS. In some scenario, the number of one-time keys generated will be kept small in order to increase signature performance and minimize signature size. In this situation, the need to use one of the one-time-keys in order to perform a PCT may significantly reduce the number of available keys for operational signatures that can be performed. Due to the factors above and in particular the potential time to generate signatures, the PCT for HBS has been simplified.

10.3.B Self-test for Embedded Cryptographic Algorithms

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

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.

10.3.C Conditional Manual Entry Self-Test Requirements

Applicable Levels:	All
Original Publishing Date:	May 4, 2021
Effective Date:	May 4, 2021
Last Modified Date:	May 4, 2021
Relevant Assertions:	AS09.15, AS10.43 - AS10.46
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

ISO/IEC 19790:2012 Section 7.10.3.5 Conditional manual entry test

If SSPs or key components are manually entered directly into a cryptographic module or if error on the part of the human operator could result in the incorrect entry of the intended value, then the following manual entry tests **shall [10.42]** be performed:

- the SSP or key components **shall [10.43]** have an error detection code (EDC) applied, or **shall [10.44]** be entered using duplicate entries.

If an EDC is used, the EDC **shall [10.45]** be at least 16 bits in length. If the EDC cannot be verified, or the duplicate entries do not match, the test **shall [10.46]** fail.

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

Directly entered (plaintext or encrypted) SSPs **shall [09.15]** be verified during entry into a module for accuracy using the conditional manual entry test specified in 7.10.3.5.

Question/Problem

When is the conditional manual entry (self) test required?

Resolution

1. The conditional manual entry test (specified in **AS10.42** - **AS10.46**) applies to:
    a. SSPs or key components that are manually entered directly.
    b. Manually entered SSPs or key components that establish authentication data, i.e. modifying SSPs within the module. The manual entry test is not applicable for operator authentication (e.g. logging into the module using established data).

No conditional manual entry test applies for SSPs that are established via automated or manual electronic entry.

Additional Comments

Please see IG 9.5.A for examples of SSPs that are either manually entered directly or electronically, or automatically entered.

10.3.D Error Logging

Applicable Levels:	3 and 4
Original Publishing Date:	August 27, 2021
Effective Date:	August 27, 2021
Last Modified Date:	August 27, 2021
Relevant Assertions:	AS10.12, AS10.13
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

ISO/IEC 19790:2012 Section 7.10.1:

At Security Levels 3 and 4, the module **shall [10.12]** maintain an error log that is accessible by an authorised operator of the module. The error log **shall [10.13]** provide information, at a minimum, the most recent error event (i.e. which self-test failed).

ISO/IEC 24759:2017:

AS10.12: (Self-tests – Levels 3 and 4) The module shall maintain an error log that is accessible by an authorised operator of the module.

VE10.12.01: The vendor documentation shall specify the error logging functionality of the module including types of recorded information in the error log (e.g. which self-test has failed, when the error occurred).

VE10.12.02: The vendor documentation shall describe the mechanism to maintain the integrity of the error log.

TE10.12.01: The tester shall verify from the vendor documentation that an unauthorised operator cannot access to the error log.

TE10.12.02: The tester shall verify from the vendor documentation that the error logging functionality provides information, at a minimum, of the most recent error event.

TE10.12.03: The tester shall cause the cryptographic module to enter an error state and verify that the module generates the error log, at a minimum, for the most recent error event.

TE10.12.04: The tester shall access the error log without assuming any authenticated role supported by the cryptographic module. If the error log can be accessed, this assertion fails.

TE10.12.05: The tester shall exercise the cryptographic module and verify that the error log is protected against unauthorised modification and substitution.

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).

IG 4.1.A Additional Comment #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.

Question/Problem

1. What constitutes an error log?
2. If the error log does not contain sensitive module information, can an authorised operator access the error log without first authenticating to the module?
3. What mechanisms are acceptable for maintaining the integrity of the error log?

Resolution

1. The intent of the error log is to allow an operator of the module to gain insight into specifically which error condition(s) occurred. This **shall** be a storage mechanism that is maintained by the module and contain at least the most recent error (i.e., which self-test failed). The error log is not required to include hard errors, which are described in Section 7.11.4 of **ISO/IEC 19790:2012**.
2. Per IG 4.1.A, 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. In the case that the error log does not contain any sensitive module information, an operator can assume an authorised role that does not require authentication in order to gain access to the module's error log.
3. Although **ISO/IEC 19790:2012** [**AS10.12**] does not explicitly require integrity protection of the error log, it does require restricted access, which implies that only authorised operators can read the contents, and the log itself cannot otherwise be modified or deleted. As such, the vendor **shall** provide documentation on how the error log is protected against unauthorised modification and substitution in VE10.12.02.

Additional Comment

1. The error log may be preserved over resetting, rebooting, or power cycling of the module, but this is not a requirement.
2. Although TE.10.12.04 indicates that authentication is required, this is in conflict with **ISO/IEC 19790:2012** [**AS10.12**] which states that an operator merely be authorised. Resolution 2 is consistent with IG 4.1.A that states an operator in the approved mode does not have to be authenticated in order to perform "show status, show version and self-tests" or other services that "do not create, disclose, modify, substitute, access, or make use of the module's CSPs, and PSPs are not modified or substituted".

10.3.E Periodic Self-Testing

Applicable Levels:	3 and 4
Original Publishing Date:	August 27, 2021
Effective Date:	August 27, 2021
Last Modified Date:	August 27, 2021
Relevant Assertions:	AS10.54, AS10.55
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

ISO/IEC 19790:2012 Section 7.10.1 states that, "Cryptographic module pre-operational and conditional self-tests provide the operator assurance that faults have not been introduced that would prevent the module's correct operation". Periodically performing the self-tests ensures that faults have not been introduced since the last time the module's self-tests were performed. At security levels 1 and 2, acceptable means for initiating the periodic self-tests include a provided service, resetting, rebooting or power cycling. At security levels 3 and 4, the module is required to contain the logic to automatically perform the pre-operational or conditional self-tests upon a defined time period.

ISO/IEC 19790:2012 Section 7.10.3.8:

SECURITY LEVELS 3 AND 4

The module **shall [10.54]** repeatedly, upon a defined time period automatically, without external input or control, perform the pre-operational or conditional self-tests.
The time period and any conditions that may result in the interruption of the module's operations during the time to repeat the pre-operational or conditional self-tests **shall [10.55]** be specified in the security policy ({**ISO/IEC 19790:2012**} Annex B) (e.g. If the module is performing mission critical services that can't be interrupted and the time period is passed for the initiation of the pre- conditional self-tests, the self-tests may be deferred after the time period is passed again).

From **ISO/IEC 24759:2017**:

TE10.54.01**:** The tester shall verify, by inspection of the cryptographic module, that the pre-operational and conditional self-tests are repeatedly performed as specified in VE10.54.01, and VE10.54.02.

ISO/IEC 19790:2012 also provides for multiple ways of performing conditional cryptographic algorithm self-tests. These can be done by known answer test, a comparison test, or a fault-detection test.

Definitions from **ISO/IEC 19790:2012** Section 7.10.3.2:

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.

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.

Question/Problem

Some modules lack certain resources such as internal clocks that make the tracking of time difficult or impossible. Other modules are only ever powered on for very short periods of time while also having extreme constraints on memory for storage or execution of code, emphasizing the need to avoid any code that isn't

going to realistically execute. A module's design may also include features that continuously protect against faults, such as comparison tests or fault-detection tests, making periodic self-testing logic redundant.

Some conditional self-tests are executed only at the time of a specific event and a repetition of that self-test will not be possible or will not be meaningful. For example, a manual entry test will make sense only at the time of the SSP entry.

1. In addition to tracking elapsed time using internal clocks within a module, are other means for establishing the time period acceptable?
2. What classes of self-tests are subject to a periodic execution? Are all types of conditional tests subject to periodic testing?
3. Are there cases where the periodic self-testing requirements are met inherently?
4. There is an apparent contradiction between the wording of **[AS10.54]** and the associated TE. The wording of **[AS10.54]** includes an "or" statement that implies that the cryptographic module vendor has an option between periodically performing pre-operational self-tests or periodically performing conditional self-tests. This interpretation gets confused when reading TE10.54.01 which implies that both pre-operational *and* conditional self-tests must be performed periodically. What is the correct interpretation?
5. If conditional cryptographic algorithm tests are to be repeated to satisfy **[AS10.54]** can execution of the algorithm self-test be delayed to the next operational use of the cryptographic algorithm? And can these tests be run in the background without the need for inhibiting data output?

Resolution

1. Defining the time period for which self-tests will be automatically invoked can be done in a variety of ways and is not dependent on a module's ability to track elapsed real time. It is up to the vendor to define the time period, but some acceptable solutions include but are not limited to clock cycles, counters, or command or operation counts.
2. One of the following classes of self-tests (a or b) are subject to a periodic execution:
   a. Pre-operational self-tests:
      i. Software/Firmware integrity test [**ISO/IEC 19790:2012** Section 7.10.2.2]
      ii. Bypass test [**ISO/IEC 19790:2012** Section 7.10.2.3]
      iii. Critical function test [**ISO/IEC 19790:2012** Section 7.10.2.4]
   b. Conditional self-tests:
      i. Cryptographic algorithm test [**ISO/IEC 19790:2012** Section 7.10.3.2]
      ii. Bypass test (inherent to the bypass test requirements) [**ISO/IEC 19790:2012** Section 7.10.3.6]
      iii. Critical function test [**ISO/IEC 19790:2012** Section 7.10.3.7]

   The other conditional self-tests (Pairwise Consistency test, Software/Firmware Load test, Manual Entry test) will only need to be performed at the time of the associated condition.
3. The periodic self-testing requirements of **ISO/IEC 19790:2012** are required by all modules even those with resource constraints. However, there are cases for which the periodic self-test requirements can be considered to be met inherently. Permitted cases (as of the publication date of this IG) are:
   a. Modules that are powered-on for short periods of time (e.g., Contactless devices only powered-on based on proximity to a reader). In this case, the module is powered on for only seconds at a time before being powered off. The self-tests are periodically performed by nature of the module in this case and additional logic is not required.

b. Modules with a maximum up-time or a limit on execution cycles. Once these limits are reached the module is forced to reset which, in turn, will invoke the module's self-tests periodically.

c. A bootloader will have a pre-operational test for firmware integrity/authenticity but in most cases is designed exclusively to launch the main firmware for the module. In this architecture, as the module isn't complete until the main firmware is launched (replacing the bootloader in executable memory), it is redundant that the bootloader itself implement a mechanism to run periodic tests - i.e., the main firmware will always be loaded and is responsible for managing the periodic self-tests where implemented.

If a vendor wishes to claim that a module meets the periodic self-testing requirements inherently based on module design or limitations and falls into one of the cases above, the vendor **shall** provide rationale in the module's security policy as to how the module is protected against faults or errors that may occur over time. More details supporting the rationale can be included in the Test Report (VE10.54.01). To make the case that a module meets the periodic self-test requirements inherently based on the amount of time the module is powered-on (case 'a' or 'b' above), the module's security policy **shall** explicitly state what the expected timeframe is for the periodic self-test.

4. It is acceptable for the cryptographic module vendor to choose which category of self-tests gets invoked periodically and it is not required that both sets of self-tests be covered by this requirement. Depending on the module's implementation, having this option may further reduce the resource burden for some modules. For example, in many cases running pre-operational self-tests can be much more efficient than running the conditional algorithm self-tests specified in Resolution 2b of this IG.

5. Yes. When the time period for the periodic self-tests lapses and conditional algorithm tests are to be repeated, it is accepted for the module to delay execution of the algorithm self-test to the next operational use of the cryptographic algorithm. In addition, conditional algorithm self-tests used for periodic tests may be run in the background without the need for inhibiting data output, unlike pre-operational self-tests which are explicitly included in **[AS03.07]** and **[AS10.03]**.

Additional Comments

1. It is possible for different self-tests to be periodically executed by different mechanisms. In the event that multiple triggers for periodic self-test are defined, each mechanism **shall** be clearly stated in the module's security policy along with the self-tests that correspond to each.

2. In the event where a module utilizes a continuous comparison test or a continuous fault detection test to satisfy the CAST requirements of **ISO/IEC 19790:2012**, these implicitly meet the periodic self-testing requirements **[AS10.54]** and no additional logic is needed.

10.3.F Complete Image Replacement Versus Software/Firmware Loading

Applicable Levels:	All
Original Publishing Date:	July 25, 2023
Effective Date:	July 25, 2023
Last Modified Date:	April 18, 2025
Relevant Assertions:	AS04.27 – AS04.35
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

Complete image replacement is an undefined term in ISO/IEC19790:2012 yet triggers two significant assertions in the form of **AS04.34** and **AS04.35** that can have a significant impact on modules and users if evoked. This implementation guidance looks to clarify when an update to the module is to be considered a complete image replacement and as such, when **AS04.34** and **AS04.35** apply.

Complete image replacement involves software/firmware loading, which itself is not well defined. Therefore, in order to clearly define complete image replacement, software/firmware loading must also be clarified. Additionally, misinterpretation of software/firmware loading has implications on **AS04.29**, thus this implementation guidance seeks to define the extent of the software/firmware loading process.

ISO/IEC 19790:2012 Terms and Definitions:

3.118 software/firmware load test: set of tests performed on software or firmware which has to pass successfully before it can be executed by a cryptographic module

NOTE: Not applicable if the software or firmware is a complete image replacement and executed only after module power cycling.

ISO/IEC 19790:2012 Section 7.4.3.4 Software/Firmware loading:

If a cryptographic module has the capability of loading software or firmware from an external source, then the following requirements **shall [04.27]** apply:

— the loaded software or firmware **shall [04.28]** be validated by a validation authority prior to loading to maintain validation;

— all data output via the data output interface **shall [04.29]** be inhibited until the software/firmware loading and load test has completed successfully;

— the Software/Firmware Load Test specified in 7.10.3.4 **shall [04.30]** be performed before the loaded code can be executed;

— the cryptographic module **shall [04.31]** withhold execution of any loaded or modified approved security functions until after the pre-operational self-tests specified in 7.10.2 have been successfully executed; and

— the modules versioning information **shall [04.32]** be modified to represent the addition and/or update of the newly loaded software or firmware (7.4.3).

If the loading of new software or firmware is a complete image replacement, this **shall [04.33]** constitute an entirely new module which would require validation by a validation authority to maintain validation. The new software or firmware image **shall [04.34]** only be executed after the module transitions through a power-on reset. All SSPs **shall [04.35]** be zeroised prior to execution of the new image.

ISO/IEC 19790:2012 Section 7.5 Software/Firmware security:

[…]

The following requirements **shall [05.03]** apply to software and firmware components of a cryptographic module for Security Level 1:

[…]

— For a software or firmware module, if the loaded software or firmware image is a complete replacement or overlay of the validated module image, the software/firmware load test is not applicable (NA) as the replacement or overlay constitutes a new module.

  If the software or firmware that is loaded is associated, bound, modifies or is an executable requisite of the validated module, then the software/firmware load test is applicable and **shall [05.13]** be performed by the validated module with the following exceptions:

  - The cryptographic module is a software module and the loaded software image is a complete image replacement or overlay of the validated module.
  - The cryptographic module is a firmware module of physical Security Level 1 and the loaded firmware image is a complete image replacement or overlay of the validated module.
  - The cryptographic module is a hybrid software module and the loaded software image is a complete image replacement or overlay of the disjoint software components.
  - The cryptographic module is a hybrid firmware module of physical Security Level 1 and the loaded firmware image is a complete image replacement or overlay of the disjoint firmware components.

[…]

Question/Problem

1. What is the definition of complete image replacement?
2. When do **AS04.34** and **AS04.35** apply?
3. What is the definition of software/firmware loading?

Resolution

1. A complete image replacement is the loading of any software/firmware components that replaces the existing module component(s) in its entirety, thus forming a completely new module that is executed without performing any software/firmware load test (described in ISO/IEC 19790:2012 Section 7.10.3.4).
2. **AS04.34** and **AS04.35** apply when a complete image replacement (as defined in Resolution #1) is supported by a module.
3. Software/firmware loading is the process of realizing first-time execution of a new or modified software/firmware component (i.e., the necessary act that a module has to perform to be able to start executing software/firmware) after the given component has been introduced into the module boundary.

Additional Comments

The nine Additional Comments below refer to software/firmware loading as opposed to a complete image replacement:

1. For software/firmware loading, all changed software/firmware components in the module must undergo the load test prior to their execution. The load test does not necessarily need to happen during: transferring these new versions to the module, saving them in the module, or writing them into the memory where they could be executed from. The load test can happen at any time as long as execution of these new versions is gated by successful completion and positive result of this test.
2. Even if the load test is always performed prior to the execution of new versions of software/firmware (and therefore a complete image replacement is not applicable), Resolution #2 does not stop a vendor from complying with **AS04.34** and **AS04.35** if they choose to do so.
3. There are at least two common situations that lead to software/firmware loading:

a. software/firmware is stored outside the module boundary:

- every time a module starts, the newly introduced software/firmware (that may be the same code each time) has to be transferred into the module, which is followed by loading and execution. In the situation where the same code is loaded each time:

1. the module is exempt from meeting **AS04.32** since the module versioning will remain unchanged;
2. **AS04.28** is implicitly met since the loaded software/firmware is part of the module to be validated and will inherit the same certificate number that will be received by the module;
3. the module **shall** meet all other requirements of Software/Firmware Load Test (**ISO/IEC 19790:2012** Section 7.10.3.4). In this case the software/firmware can be loaded during initialisation as an exception to Additional Comment 9 in this IG.

b. software/firmware is stored inside the module boundary, but an operator initiates the process of changing the software/firmware from the current version to a new version that is transferred into the module (it is often called a software/firmware update) followed by requesting execution of this new version (it is often done by restarting the module or invoking a dedicated operation):

- loading, as defined in Resolution #3, happens only before the first execution of the new software/firmware version (and not before subsequent executions).

4. Transferring software/firmware into the module (and potentially saving it there for future use) is not considered part of the software/firmware loading, unless the transferring itself constitutes software/firmware loading as defined in Resolution #3.

5. An example of steps that realizing first-time execution of software/firmware can consist of: preparing for the lack of software/firmware, stopping current software/firmware execution (if there is any), copying new software/firmware into executable memory (if it is not already there), performing software/firmware load test for new software/firmware (if not already done and no complete image replacement claimed), any other module specific activity necessary before execution becomes feasible.

6. For the purposes of FIPS 140-3, the bitstream used to configure an FPGA is considered as firmware. Although not executing on traditional CPU, the bitstream contains a series of instructions parsed by the FPGA on start-up to configure its internal logic and where this can be considered as a highly customized form of executable code. Therefore, the firmware load, as defined in Resolution #3 is applicable when loading the bitstream within the module's boundary, unless it constitutes a complete image replacement, per Resolution #1. Also, the software/firmware integrity test is required for an FPGA that stores the bitstream persistently within its boundary (e.g., the integrity test could be an EDC on a hardware module per **AS05.05** or also be the software/firmware load test as per Additional Comment #8 of this IG).

7. Scenario 1: Software/firmware may contain a reference authentication key [**AS10.38**] for use in verifying future software/firmware, after this software/firmware has been successfully verified with the previously loaded reference authentication key.

   Scenario 2: It may also be the case where a trusted anchor is previously loaded within the existing software/firmware and is used to verify (using an approved integrity technique that may include a certificate chain) the reference authentication key that will be used to verify (using an approved data authentication technique) the newly loaded software/firmware. In this case, the reference authentication key can be loaded at the same time as the new software/firmware it will verify.

   In either of these scenarios TE10.37.09 can be verified by the lab either:

   a. with a test-firmware which does not contain or has corrupted the initial reference authentication key or trust anchor; OR

a. by modifying / removing on the fly (e.g., debugger) the previously loaded reference authentication key or trust anchor that is already in the running software/firmware

8. If software/firmware load test is to be considered inherently performed by performing software/firmware integrity test after introducing software/firmware into the module, all the following conditions **shall** be met:

   a. the software/firmware load test and the software/firmware integrity test use the same cryptographic algorithm that meets the requirements of an approved data authentication technique; e.g., EDCs are not allowed in this case as a software/firmware integrity test.

   b. the cryptographic algorithm self-test has been already conducted and successfully completed for this cryptographic algorithm (see IG 10.2.A);

   c. the software/firmware integrity test authenticates all software/firmware components introduced;

   d. and software/firmware introduced into the module cannot be executed until software/firmware integrity successfully passes.

   Even when the software/firmware load test is inherently met, the module must still meet requirements of software/firmware loading, including TE10.37.07 for the reference authentication key.

9. Software/firmware loading **shall** only occur as part of an approved service. During module initialisation (i.e., before the module moves to the operational state) or as part of a non-approved service, the module cannot execute the load test approved algorithm and SSPs, and would not be able to meet all the requirements of **ISO/IEC 19790:2012** 7.4.3.4 *Software/Firmware loading*. For that reason, Software/firmware loading **shall** **NOT** occur during module initialisation (i.e., before the module moves to the operational state) or as part of a non-approved service.

Section 11 – Life-cycle assurance

11.A CVE Management

Applicable Levels:	All
Original Publishing Date:	May 4, 2021
Effective Date:	May 4, 2021
Last Modified Date:	May 4, 2021
Relevant Assertions:	AS04.13, AS11.38
Relevant Test Requirements:	TE04.13.02, TE11.38.03
Relevant Vendor Requirements:	VE04.13.02, TE11.38.03

Background

It is important for users, such as government agencies, and vendors be able to mitigate or eliminate vulnerabilities from their systems. It is also important that vendors be aware of known vulnerabilities within their systems and be supportive of users. The goal of CMVP is to provide secure communications and storage. In order to support both users and vendors, CMVP is requiring vendors to use the National Vulnerability Database (NVD) to better assure user's acquisition and use of secure validated equipment. CMVP will support vendor management of Common Vulnerabilities and Exposures (CVE®) mitigation during and after validation.

CMVP is not requiring that the module and associated equipment be free from CVEs but is requiring that vendors use this as one of their tools to provide a more secure product to the end user. **ISO/IEC 24759:2017** requires the vendor report use of functional and automated security diagnostic tools.

AS11.29: (Vendor testing — Levels 1, 2, 3, and 4)

Documentation shall specify the functional testing performed on the cryptographic module.

AS11.30: (Vendor testing — Levels 1, 2, 3, and 4)

For software or firmware cryptographic modules and the software or firmware component of a hybrid module, the vendor shall use current automated security diagnostic tools (e.g. detect buffer overflow).

AS11.38: (Guidance documents — Levels 1, 2, 3, and 4)

***Administrator guidance* shall specify:**

- **the administrative functions, security events, security parameters (and parameter values, as appropriate), physical ports, and logical interfaces of the cryptographic module available to the Crypto Officer and/or other administrative roles;**
- **procedures required to keep operator authentication data and mechanisms functionally independent;**
- **procedures on how to administer the cryptographic module in an approved mode of operation; and**
- **assumptions regarding User behavior that are relevant to the secure operation of the cryptographic module.**

The National Vulnerability Database includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics. The NVD supports an API for CVE and CPE searching capabilities. The CPE permits a way for CVE platforms to be delineated. These resources are available to module operators to aid in the security awareness and operation of the module.

The Common Vulnerabilities and Exposures (CVE®) is a dictionary of publicly disclosed cybersecurity vulnerabilities and exposures which can be found at https://cve.mitre.org/index.html. CVE defines a vulnerability as:

"A weakness in the computational logic (e.g., code) found in software and hardware components that, when exploited, results in a negative impact to confidentiality, integrity, or availability. Mitigation of the vulnerabilities in this context typically involves coding changes but could also include specification changes or even specification deprecations (e.g., removal of affected protocols or functionality in their entirety)."

The vendors work with CVE Naming Authorities (CNA) to address vulnerabilities identified by the vendor, users, supply chain members, or secure community researchers. This is forwarded to the CVE List. The CVE List feeds NVD (https://nvd.nist.gov/), which then builds upon the information included in CVE entries to provide enhanced information for each entry such as fix information, severity scores, and impact ratings. As part of its enhanced information, NVD also provides advanced searching features such as by OS; by vendor name, product name, and/or version number; and by vulnerability type, severity, related exploit range, and impact. All vulnerabilities in the NVD have been assigned a CVE identifier and thus, abide by this definition.

The Common Platform Enumeration (CPE) Dictionary is a structured naming scheme for information technology systems, software, and packages. Based upon the generic syntax for Uniform Resource Identifiers (URI), CPE includes a formal name format, a method for checking names against a system, and a description format for binding text and tests to a name. CPEs can be used by the supply chain to further identify module instances that may be subject to a particular vulnerability.

SP 800-140 supplements the **ISO/IEC 24759:2017** vendor and test requirements for **AS11.38** to include vendors of all cryptographic modules to assess known vulnerabilities within the module and identifying any issues that users should be aware of, along with mitigation strategies if available.

VE11.38.03: The vendor shall list any CVEs associated with the module, and either

- provide assurance that they are not security relevant;
- or, for any CVE's that are considered security relevant, the vendor shall describe how they have been mitigated.

TE11.38.03: The tester shall verify that any CVEs associated with the module:

- are not security relevant, or,
- if they are security relevant, mitigations provided by the vendor are appropriate.

Question/Problem

What are the responsibilities of the cryptographic module vendor and CST Laboratory regarding how to address CVEs during and after FIPS 140-3 validations?

Resolution

<u>During IUT</u>

1. A vendor's responsibilities are:
    a. Tracking and providing a list of all security relevant and non-security relevant CVEs associated with the module or module components.
    b. Addressing the applicability of each CVE to the module.
    c. Addressing the remediation if necessary, including working with NVD to define CPEs as needed.
2. A CST laboratory's responsibilities are:
    a. review the CVE related evidence produced by the vendor and make an independent determination as to whether all applicable CVE(s) are adequate and reasonable.

b. work with the vendor to address issues discovered. The submission should also include any unresolved issues.

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.

During Review Pending, In Review, Coordination, Finalization

If new security relevant CVEs associated to the module are published after the module report is submitted, the vendor **shall** document each applicable CVE with, at minimum, the following information:

a. CVE identifier.

b. CPE identifiers if available.

c. Detailed description of the impact of the CVE to the cryptographic services provided by the module.

d. A mitigation plan of when and how to address each CVE.

e. A rationale to justify the CVE mitigation plan.

The vendor **shall** notify the CST laboratory and provide the updated information. The CST laboratory will provide this information to the CMVP through an update to the Test Report (e.g. TE11.38.03).

The lab **shall** facilitate communications between the CMVP and vendors about applicable CVEs throughout the validation process. In the event of a disagreement of resolution between CST lab and vendor, the CMVP **shall** be notified.

The final decision on whether a certain CVE must be addressed before the FIPS 140-3 certificate issuance remains with the CMVP.

Post-validation

If a CVE is discovered after FIPS 140-3 validation, it may require module design and/or implementation modifications, updates to procedural guidance, or both. The vendor **shall** take actions as deemed necessary for the module to remain on the list of FIPS 140-3 validated modules. Notification by the vendor, preferably through the point of contact, directly to CMVP may be helpful; however, the vendor must work with a CST lab for any updated submissions.

If a vendor independently discovers or is made aware of a vulnerability requiring changes to one or more of its validated modules, the vendor **shall** notify CMVP and work with NVD to identify and post a new CVE and CMVP to address any necessary changes.

Additional Comments

1. Vendors should submit any updated CVE information with any scenario that affects the module. Scenarios that only affect the vendor information do not require update CVE information.
2. In the event of a security relevant CVE discovered after validation, only FS, UPDT, and CVE scenarios will be accepted by the CMVP for any revalidations. Please see the Management Manual 7.1 *Submission Scenarios* & Annex A.2 *Addressing CVE Relevant Vulnerabilities* for more detail.

Section 12 – Mitigation of other attacks

12.A Mitigation of Other Attacks

Applicable Levels:	All
Original Publishing Date:	May 4, 2021
Effective Date:	May 4, 2021
Last Modified Date:	May 4, 2021
Relevant Assertions:	AS12.02
Relevant Test Requirements:	TE12.02.01
Relevant Vendor Requirements:	VE12.02.01

Background

ISO/IEC 19790:2012 Section 7.12:

If a cryptographic module is designed to mitigate one or more specific attack(s) not defined elsewhere in {**ISO/IEC 19790:2012**}, then the module's supporting documents **shall [12.02]** enumerate the attack(s) the module is designed to mitigate.

Question/Problem

When is this section applicable?

Resolution

If a cryptographic module has been *purposely* designed, built and publicly documented to mitigate one or more specific attacks, this section is applicable and **AS12.01** through **AS12.04** (as applicable) **shall** be addressed regardless if the vendor of the module wishes to address the claim or not. Mitigation mechanisms may address both invasive (physical) or non-invasive mechanisms. The testing laboratory, upon inspection of the modules design and documentation (both proprietary and public), **shall** verify the implemented mitigation mechanisms and/or mitigations claimed by the vendor as specified in **AS12.01** through **AS12.04** (as applicable).

Example: FIPS 140-3 Section 7.7 Level 2 is claimed. However, the vendor states that module design includes a switch that will cause zeroisation of unprotected SSPs if some part of the module is opened or penetrated. Since this is not required at Level 2, for the vendor to claim this feature, it **shall** be addressed in FIPS 140-3 Section 7.12 as an additional mitigation mechanism.

Additional Comments

Until requirements of **SP 800-140F** are defined, non-invasive mechanisms fall under **ISO/IEC 19790:2012** Section 7.12 *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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	March 26, 2024
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

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 800-56Arev3** 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 800-56Arev3** 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 800-56Arev3**.

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, brainpoolP384r1, and brainpoolP512r1.

    The allowance for the Brainpool curves also includes the twisted variants of each curve size (i.e., brainpoolP224**t**1, brainpoolP256**t**1, brainpoolP320**t**1, brainpoolP384**t**1, and brainpoolP512**t**1) (defined in RFC 5639) assuming the module verifies the domain parameter validity to ensure the correct curves are used (i.e., not its quadratic twist). Approved methods for obtaining this assurance are provided in **SP 800-89** (also see **SP 800-186**, Appendix D.1).

    Until CAVP provides testing for these Brainpool curves and their twisted variants, these can only be used in the approved mode within ECDSA or key agreement schemes as *allowed,* if all the requirements of this IG are met.

b. (From **SP 800-186** Appendix H.2) The curve secp256k1 specified in SEC 2: Recommended Elliptic Curve Domain Parameters, which supports a security strength of 128 bits. This curve is a Koblitz curve with coefficients selected for efficiency reasons. The curve secp256k1 is allowed to be used for blockchain-related applications.

   To use secp256k1 in the approved mode:

   1. This curve **shall** be used within ECDSA as *allowed* per the requirements of this IG. It **shall** not be considered *approved* unless CAVP testing is available.
   2. The module **shall** support at least one CAVP-testable curve so that the same ECDSA that uses secp256k1 can have an associated CAVP certificate.
   3. The module's Security Policy **shall** state this curve may only be used in blockchain-related applications. For more information on blockchain technology, see NISTIR 8202.
   4. The testing lab **shall** verify compliance to SEC 2 Sections 2 and 3, as applicable, through code review and testing the module's functionality.

2. During the transition period (see IG C.K) 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:

a. the algorithm implementation **shall** use approved underlying algorithms, such as the message digests,

b. the Security Policy **shall** list all approved and non-approved curves that are implemented,

c. the Security Policy **shall** indicate the associated security strength for all non-approved curves that are implemented.

d. [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

e. if ECDSA or KAS (using elliptic curve cryptography) is listed on the certificate's non-approved but allowed line, the algorithm implementation **shall** have been CAVP tested and validated for at least one approved elliptic curve.

Additional Comments

1. EdDSA in **FIPS 186-5** **shall** only be used with the specified curves in **FIPS 186-5** and **SP 800-186** (i.e., Edwards25519 and Edwards448). Approved services **shall** not use EdDSA with the Brainpool curves in Category 1 of this IG.
2. See IG C.K for the transition guidance regarding **FIPS 186-5** and **SP 800-186** and any impact it may have on validated module status.

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

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

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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	October 21, 2024
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

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. Outside of **FIPS 186-5** and **SP 800-208** specifications, 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 7.2 *CMVP requirements pertaining to testing and approved algorithms* is applicable.

Additional Comments

1. None.

C.D Use of a Truncated HMAC

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	September 21, 2020
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01, TE02.20.02
Relevant Vendor Requirements:	VE02.20.01, TE02.20.02

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

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 RSA Approved Parameter Sizes in FIPS 186-5

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	July 26, 2024
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

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 800-131Arev2**, the 1024-bit RSA modulus may be used, for legacy purposes, in the approved mode for signature verification, but is disallowed for signature generation.

In February 2023 NIST published **FIPS 186-5**. This standard replaces **FIPS 186-4**; however, both standards have been in effect through February 3, 2024. **FIPS 186-5** has approved the RSA modulus sizes of at least 2048 bits with no upper limit.

Question/Problem

Several NIST standards and the CMVP Guidance imply various restrictions on the sizes of the RSA moduli approved for use in the RSA signature generation and the RSA signature verification algorithms. **FIPS 186-4** approves the use of three modulus sizes: 1024, 2048, and 3072 bits. There is also an older standard, **FIPS 186-2**, which defers to an ANSI standard **X9.31**. The latter standard permits the use of the RSA modulus sizes that can be represented as 1024 + 256*s bits, where s is a non-negative integer.

The algorithm transition standard **SP 800-131Arev2** approves, for signature generation, all RSA modulus sizes of at least 2048 bits. Finally, **FIPS 186-5** permits an RSA signature generation and/or verification modulus bit size to be any integer no smaller than 2048.

In view of these restrictions placed into different documents, which modulus sizes can be used in the approved mode (a) when generating the RSA signatures and (b) when verifying the RSA signatures? What sizes of the RSA auxiliary primes defined in Appendix A.1.1 of **FIPS 186-5** are acceptable for a given size of the RSA primes whose product produces an RSA modulus?

What is the minimum number of the Miller-Rabin tests used when performing the probabilistic primality testing? Shall the method of Appendix C of **FIPS 186-5** for computing this number be followed?

Resolution

Signature and key generation. When performing an RSA signature generation in compliance with **FIPS 186-5**, 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. This guarantees that CAVP testing can be performed with at least one implemented RSA modulus size.

The RSA signature algorithm implementations **shall** be tested by an accredited testing 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. If no CAVP testing is available for the RSA signature generation with a particular approved RSA modulus size implemented in the module (e.g., 2304, 2560 or 16384 bits), then the untested status of these modulus sizes **shall** be documented in the Security Policy.

Some of the RSA key generation methods described in Appendix A of **FIPS 186-5** 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 A.1 in **FIPS 186-5** specifies the minimum bit lengths and the maximum total lengths of the auxiliary primes. The minimum numbers of the Miller-Rabin tests used in primality testing **shall** be consistent with the

bit sizes of p, q, p_1, p_2, q_1 and q_2 . Vendors **shall** use at least as many Miller-Rabin tests with the randomly generated bases, as what is shown in Table B.1 of **FIPS 186-5**. The entries in this table have been computed by a method presented in Appendix C.1 of **FIPS 186-5**. However, as explained in a Warning at the end of that Appendix, the proposed approach might be too aggressive and a more cautious methodology for computing the minimum required number of the Miller-Rabin tests may be warranted. This Implementation Guidance *strongly recommends, but does not require* that in the modules claiming compliance with **FIPS 186-5** the vendor implements one of the following two solutions when deciding on the minimum required number of the Miller-Rabin tests while testing the probabilistic primality of the RSA primes p and q:

(1) At the minimum, perform the number of the Miller-Rabin tests indicated in the middle column in Table B.1 of **FIPS 186-5** (e.g., two tests with the randomly chosen Miller-Rabin bases (following an algorithm either in section B.3.1 or section B.3.2 of **FIPS 186-5**) when the size of a candidate prime is 2048 bits) followed by the Lucas test defined in section B.3.3 of **FIPS 186-5**.

(2) Perform at least 25 Miller-Rabin tests with the randomly chosen Miller-Rabin bases.

See an Additional Comment below for a more detailed justification for this recommendation.

If the bit size of an RSA prime falls between the numbers shown in Table B.1 of **FIPS 186-5** (i.e., that size is not equal to 1024, 1536 or 2048) then the number of the Miller-Rabin tests defined above **shall** be that of the next smaller size included in the table. That is, if the length of p and q is 1792 bits then the rules for the 1536-bit primes apply. If the lengths of p and q are greater than 2048 bits, then apply the rules for the 2048-bit primes.

The minimum number of the Miller-Rabin tests when testing the primality of p_1, p_2, q_1 and q_2 is also provided in Table B.1 of **FIPS 186-5**. For these auxiliary prime candidates, the entries in Table B.1 of **FIPS 186-5** are fully adequate and it is not recommended that the additional Miller-Rabin tests or the Lucas test are performed. As before, if the size of p and q falls between the numbers shown in Table B.1 then the number of the Miller-Rabin tests required for the auxiliary primes associated with the smaller p and q will be applicable: if the lengths of p and q are 1280 bits then each of p_1, p_2, q_1 and q_2 **shall** undergo at least 32 rounds of the Miller-Rabin testing. The restrictions on the sizes of the auxiliary primes are shown in Table A.1 of **FIPS 186-5**.

The use of the approved hash functions in digital signatures is documented in **SP 800-131Arev2,** Table 8. The choice of a hash function may affect the security strength of the RSA signature algorithm.

Signature verification. When performing a **FIPS 186-2** signature verification (for legacy use), the module may support all modulus bit sizes shown in **FIPS 186-2**; that is, all numbers in the form of 1024+256*s, where s is a nonnegative integer.

When performing a **FIPS 186-4** signature verification (for legacy use), the module may support all RSA modulus sizes no smaller than 2048, as well as the 1024-bit modulus. For the **FIPS 186-5** signature verification, the modulus size **shall** be at least 2048.

If the module performs the **FIPS 186-2** (for legacy use)**, FIPS 186-4** (for legacy use) or **FIPS 186-5** RSA signature verification, then this algorithm **shall** be tested by the CAVP with at least one modulus size for compliance with this specific version of FIPS 186. All modulus sizes where testing is available **shall** be tested by the CAVP. If no CAVP testing is available for the RSA signature verification with a particular approved RSA modulus size, then the untested status of these modulus sizes **shall** be documented in the Security Policy.

Additional Comments

1. The bounds on the sizes of the auxiliary primes p_1, p_2, q_1 and q_2 are given in Table A.1 of **FIPS 186-5**. Note that the upper bounds for the values of len(p_1) + len(p_2) and len(q_1) + len(q_2) for both the probable and the provable primes p and q are the sanity-check constraints. If p_1 and p_2 were large enough to exceed the stated upper bound for len(p_1) + len(p_2) then there would be very few possibilities left (in some cases, none) for choosing the prime number p. The CMVP is not aware of any security rationale for generating the auxiliary primes p_1, p_2, q_1 and q_2 greater than their minimum length stated, as a function of *nlen*, in Table A.1 of **FIPS 186-5**. Hence the CMVP does not encourage vendors to use the auxiliary primes whose lengths exceed the minimum stated values.

2. The Miller-Rabin primality test numbers entries in Table B.1 of **FIPS 186-5** are populated by applying an algorithm from Appendix C.1 of **FIPS 186-5.** Note that of the most interest to the

developer of a cryptographic module is not the probability that a composite number taken among all integers of a given size passes *t* randomly-selected rounds of the Miller-Rabin tests (this probability becomes low as the pool of the prime candidates increases fast when their bit lengths increase) but the probability that a given number passing *t* rounds of the Miller-Rabin tests with randomly chosen bases is composite.

When computing the latter probability, no averaging over all numbers of the same bit size takes place. Since the probabilities of the different composites of the same size to pass the Miller-Rabin tests may be very different, this averaging can be misleading. However, the result of this averaging is what is used when computing the minimum required number of the Miller-Rabin tests in Appendix C.1 of **FIPS 186-5**. To mitigate the risks of accidentally letting a composite value of *p* or *q* pass the tests it is recommended in this Implementation Guidance that the user implements either the Lucas test (in addition to the minimum number, as shown in Table B.1 of **FIPS 186-5**) of the Miller-Rabin tests or increases the number of the Miller-Rabin tests to at least 25.

The reason behind the 25 Miller-Rabin tests with randomly chosen bases when selecting Option (2) recommended in the **Resolution** section of this IG, is that it is a heuristic compromise between overly ambitious and not fully justified **FIPS 186-5** requirements (such as, only two M-R tests for the 2048-bit-long *p* and *q* when accepting a 2^{-100} probability of generating a composite) and the substantiated but overly conservative 50 M-R tests recommendation if computing this number as $\lceil -\log_2(p_{\text{target}})/2 \rceil$, where p_{target} is, again, 2^{-100}.

C.G Moved to W.3

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

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	October 21, 2024
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

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 Scenarios 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, 5289, and 9325 provisions; DTLS 1.2 GCM Cipher Suites for DTLS, as described in RFCs 5116, 6347, and 9325 provisions.
    b. IPsec-v3 protocol, as described in RFCs 4106, 5282, and 7296.
    c. MACsec with GCM-AES-128, GCM-AES-256, GCM-AES-XPN-128 and GCM-AES-XPN-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/DTLS 1.2 protocol IV generation

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

    If the vendor claims that the IV generation is in compliance with the TLS/DTLS 1.2 specification and only for use within the TLS/DTLS 1.2 protocol, then the module's Security Policy and the Validation Test Report **shall** explicitly state the module's compatibility with TLS/DTLS 1.2 and the module's support for acceptable AES-GCM ciphersuites from Section 3.3.1 of **SP 800-52 rev1** or **SP 800-52rev2**.

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

i) The operations of one of the two parties involved in the TLS/DTLS 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/DTLS 1.2 protocol implementation that relies on the module being validated against an independently developed instance of TLS/DTLS 1.2, such as the many TLS/DTLS 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 in 3a) **shall** be 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 (e.g., 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/DTLS 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 3a 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 3a of this Implementation Guidance.

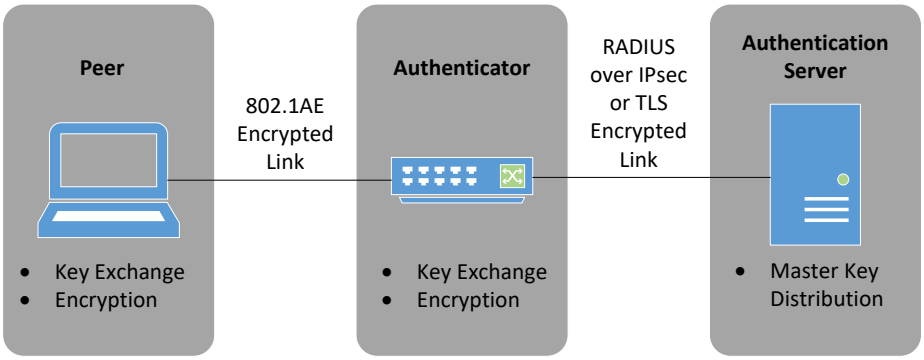


Figure 1. MACsec protocol components

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 3a 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 that is internal to the module's boundary and

- 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.

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 2^{32} 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. If it is demonstrated in the Test Report that this exhaust condition is guaranteed to never be reached, this will exempt the module from needing to implement an abort logic. (Please note that the following 3a also applies for this scenario 3).

   a. 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 industry [1] 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.
   - 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.

[1] 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).

- 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. In Scenario 2 it is recommended but not required that the entropy source producing the DRBG seed is located inside the module's boundary. The reason for not making this a firm requirement is that even if little or no entropy is supplied to the DRBG, this **SP 800-90A**-compliant random bit generator will be generating the non-repeating outputs. The probability of producing the matching DRBG outputs depends on the details of the design of the DRBG, as shown in **SP 800-90A**, but in all cases this probability is much lower than 2^{-32}. Therefore, the probability of generating the matching (key, IV) pairs, which is no greater than the probability of generating the matching IV parameters, is less than 2^{-32}, if the IVs are generated by the same module.

   Are there any potential vulnerabilities in a scheme that does not require an internal (to module's boundary) generation of the DRBG seed by a SP 800-90B-compliant entropy source? Yes. For example, if the module's DRBG is seeded with the same entropy each time the module gets restarted or instantiated, *and* the same sequence of generating keys and IVs is repeated during separate restarts/instantiations after the module receives the initial DRBG seed, *and* the DRBG does not retain its state between the restarts, the module may generate the identical (key, IV) pairs that will be used in separate AES GCM encryptions.

In a different scenario, if two modules using the same AES GCM encryption key receive an identical DRBG seed from a third party *and* each module is using the DRBG outputs in the same order as the other module until each module generates an IV, then two separate AES GCM encryptions may be performed with the same (key, IV) pair.

The CMVP considers these scenarios to be far-fetched and significantly less likely to occur than receiving a poorly generated key from outside the module's boundary. Hence, generating entropy inside the module's boundary, while recommended, is not required. The entropy caveats shown in IG 9.3.A must be documented in the module's certificate, as applicable.

5. Note that while the entropy may be generated externally, the requirement that an approved and tested DRBG located within the module's boundary is used to generate the AES GCM IVs must be met in Scenario 2.
6. 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.
7. 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.
8. 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.
9. 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 (SP 800-38E) Requirements on the Key

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	Month day, year
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

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 800-133rev2**, 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 800-133rev2**, 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	September 21, 2020
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

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**.

C.K Transition from FIPS 186-4 to FIPS 186-5 and SP 800-186

Applicable Levels:	All
Original Publishing Date:	July 25, 2023
Effective Date:	July 25, 2023
Last Modified Date:	April 18, 2025
Transition End Dates	**February 4, 2024 – See Below**
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

Background

FIPS 186-5, Digital Signature Standard (DSS), along with the companion document **SP 800-186**, *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*, was published on February 4, 2023, to supersede **FIPS 186-4**. This IG provides the details of a transition from **FIPS 186-4** to **FIPS 186-5** and **SP 800-186**.

FIPS 186-5 specifies three approved techniques for the generation and verification of digital signatures that can be used for the protection of data:

1. Rivest-Shamir-Adleman (RSA) Algorithm, as specified in PKCS #1 (i.e., RSASSA-PKCS1-v1.5 and RSASSA-PSS). **FIPS 186-5** also introduces larger modulus sizes for RSA (with updated tables A.1 and B.1). ANSI X9.31 was withdrawn from **FIPS 186-5**, and therefore X9.31 RSA signatures were removed.
2. Elliptic Curve Digital Signature Algorithm (ECDSA). The reference to ANSI X9.62 was removed, so the specifications for ECDSA were added directly into **FIPS 186-5**. A variant of ECDSA with a deterministic signature generation procedure (known as deterministic ECDSA), as specified in IETF RFC 6979, is also approved. Recommended elliptic curves for Federal Government use of ECDSA (including deterministic ECDSA) are provided in **SP 800-186**.
3. Edwards Curve Digital Signature Algorithm (EdDSA), as specified in IETF RFC 8032. **FIPS 186-5** approves the use of EdDSA and specifies additional requirements. Recommended elliptic curves for Federal Government use of EdDSA are provided in **SP 800-186**. Also included is HashEdDSA, a version of EdDSA where the EdDSA signature is generated on the hash of the message rather than the message itself.

An approved Extendable-Output Function (XOF) specified in **FIPS 202** is now approved within RSA-PSS and ECDSA; before this could only be an approved hash function (e.g., SHA2, SHA3).

The Digital Signature Algorithm (DSA), which was specified in prior versions of **FIPS 186**, is no longer specified in **FIPS 186-5** and may only be used to *verify* existing signatures. Complete specifications of DSA can be found in **FIPS 186-4**.

SP 800-186 specifies the set of recommended elliptic curves. In addition to the previously recommended Weierstrass curves (i.e., Curves P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, and B-571), there are two newly specified Edwards curves (i.e., Edwards25519, Edwards448) included for use with the EdDSA algorithm. While **SP 800-186** includes the specifications for elliptic curves over binary fields (i.e., K-233, K-283, K-409, K-571, B-233, B-283, B-409, and B-571), the use of these curves for the ECDSA signature generation is now deprecated, and the use of other (prime) curves (i.e., P-224, P-256, P-384, P-521) is strongly recommended. Also, the option to generate elliptic curves (besides those specified in **SP 800-186**) is removed. Similarly, users are not given the option to generate their own base points on elliptic curves.

Question/Problem

What is the CMVP transition schedule regarding the validation to the **FIPS 186-4**, **FIPS 186-5** and **SP 800-186** standards?

Specifically, for:

1. DSA and X9.31 RSA which have both been removed from **FIPS 186-5**.
2. Curves permitted under IG C.A for ECDSA and ECDH but not permitted in **SP 800-186**.
3. **FIPS 186-4** primes and generator (Appendix A), and FIPS 186-type FFC key-pair generation, which was not included in **FIPS 186-5**.
4. Curves defined over binary fields since these were deprecated in **SP 800-186**.
5. Curves permitted under **SP 800-56Arev3**.
6. Signature verification when the corresponding signature generation is no longer approved / allowed.

Resolution

1. On **Feb 5, 2024**, the CMVP will no longer accept new submissions that implement DSA or RSA X9.31 in the approved mode, other than for signature verification which is still approved for legacy purposes for both algorithms.
2. On **Feb 5, 2024**, the CMVP will no longer accept new submissions that implement curves permitted under IG C.A for ECDSA and ECDH but not included in **SP 800-186** for use in the approved mode (i.e., IG C.A Category 2). Until this date, these will continue to be accepted as *allowed* per IG C.A.
3. The CMVP will continue to accept new submissions that implement **FIPS 186-4** primes and group generators obtained per **FIPS 186-4** Appendix A *Generation and Validation of FFC Domain Parameters* used exclusively in an **SP 800-56Arev3**-compliant scheme. **SP 800-56Arev3** refers to these as "FIPS 186-type" domain parameters. This allowance also applies to the key-pairs generated using these FIPS 186-type FFC parameter-size sets (**SP 800-56Arev3** Sec. 5.6.1.1.2) within an approved key-agreement scheme. The corresponding CAVP tests are DSA PQGGen and DSA KeyGen. IG D.F may later define a transition for these.
4. There is currently no scheduled transition away from elliptic curves over binary fields (i.e., K-233, B-233, K-283, B-283, K-409, B-409, K-571, B-571). However, these curves are now deprecated, and it is *strongly recommended* to use the **SP-800-186**-defined prime curves (i.e., P-224, P-256, P-384, P-521) for the generation of the ECDSA signatures. Despite their deprecation status, these curves are still considered approved and therefore will be included in the CAVP **FIPS 186-5** testing.
5. The publication of **SP 800-186** does not impact the curves permitted under **SP 800-56Arev3**. Curves that are included in **SP 800-186** but not included in **SP 800-56Arev3** are not approved for key agreement. E.g., the ECDH X25519 and X448 key agreement schemes (defined in RFC 7748) that use Curve25519 and Curve448, respectively, are not compliant to **SP 800-56Arev3**. The corresponding Montgomery curves Curve25519 and Curve448 are included in **SP 800-186** only for use within digital signatures as alternative representations of the Edwards25519 and Edwards448 curves, as shown in Table 2 of **SP 800-186**.
6. In addition to the approved digital signature verification methods specified in **FIPS 186-5**, digital signature verification is approved for all versions of the RSA, DSA, and ECDSA defined in earlier versions of **FIPS 186** (e.g., DSA, RSA X9.31). If the keys provide 80 to 111 bits of security strength, they are only permitted for *legacy use*. After the transition date specified in Resolution 1 above, DSA and RSA X9.31 with strength 112-bits or more will also only be permitted for *legacy use*. Digital signature verification is allowed for the following algorithm: ECDSA using curves permitted by IG C.A that are not one of the curves recommended in **SP 800-186** (for *legacy use* after the transition date specified in Resolution 2 above). There is no scheduled transition for any of these digital signature verification algorithms as these continue to be approved or allowed for verifying already existing signatures.

Additional Comments

1. All transitions mentioned in this IG are 'soft' transitions in that they will not move modules to the Historical List on their respective transition dates.

2. After the transition dates of this IG (i.e., Feb 5, 2024), the CAVP will continue to provide testing for **FIPS 186-4** algorithms that **shall** only be used in a module submission where testing **FIPS 186-4** algorithms (i.e., DSA or RSA X9.31) are necessary. For example:

    a. DSA and RSA X9.31 testing may be necessary for submissions that do not need to meet the latest guidance (such as a OEUP per FIPS 140-3 Management Manual 7.1.7 that would require retesting the **FIPS 186-4** algorithm on the new environments).

    b. DSA Key_Gen and DSA PQG_Gen used solely as part of an approved **SP 800-56Arev3** FFC scheme.

    c. Legacy DSA or RSA X9.31 SigVer.

    Otherwise (i.e. in cases where testing **FIPS 186-4** algorithms are not necessary), CAVP tests done Feb 5, 2024, or later **shall** be **FIPS 186-5** tests as opposed to **FIPS 186-4** tests (even if CAVP still offers **FIPS 186-4** tests that are mathematically identical to the **FIPS 186-5** tests per Additional Comment #3 below).

3. For **FIPS 186-4** CAVP tests done prior to the Feb 5, 2024, and that are mathematically identical to **FIPS 186-5** CAVP tests, a module submission may claim **FIPS 186-5** compliance for these tests. As of the Last Modified Date of this IG, this allowance is applicable to all **FIPS 186-4** tests that are also part of the **FIPS 186-5** tests (e.g., **FIPS 186-4** tests other than DSA and RSA X9.31).

4. The CAVP algorithm validation will indicate if the algorithm was tested in compliance to **FIPS 186-5** or **FIPS 186-4**. The **FIPS 186-5** tests will not include algorithms that have been removed (e.g., DSA, RSA X9.31).

5. The term "new submissions" in this IG is referring to the submissions that are required to meet the latest CMVP guidance at the time of submission (e.g., FS, UPDT). See FIPS 140-3 Management Manual Section 7.1 for more details.

6. It is strongly recommended for modules submitted to the CMVP to comply with **FIPS 186-5** and **SP 800-186**, even before the transition dates specified in this IG. See IG 10.3.A for the associated self-test requirements.

7. Section A.3.3 of **FIPS 186-5** says that for the deterministic ECDSA, the same hash must be used for the HMAC_DRBG and signing the message. Since **SP 800-90Arev1** does not allow SHAKE128 or SHAKE256 within HMAC_DRBG, SHAKE is not permitted for signing the deterministic ECDSA message, even though **FIPS 186-5** Section 6.4 says an approved XOF (i.e., SHAKE128 or SHAKE256) is an option for the deterministic ECDSA.

8. **FIPS 186-5** specifies that for EdDSA with curve Edwards25519, SHA-512 (**FIPS 180-4**) must be used, and for EdDSA with curve Edwards448, SHAKE256 must be used. No other hash or XOF functions are permitted. This was intentional and to remain consistent with IETF RFC 8032 for interoperability.

9. Federal Register Notice (2023-02273) says, "To facilitate a transition to the new standard, **FIPS 186-4** will remain in effect alongside **FIPS 186-5** for a period of one year." This IG is consistent with this one-year transition specified for **FIPS 186-4**.

10. **SP 800-186** Table 2 says that the following curves are not to be used for ECDSA or EdDSA directly: Curve25519, W-25519, Curve448, E448, W-448. **SP 800-186** B.1 explains: "Implementations may take advantage of these mappings to carry out elliptic curve group operations that were originally defined for a twisted Edwards curve on the corresponding Montgomery curve, or vice-versa, and translating the result back to the original curve to potentially allow code reuse". This is permitted in the approved mode, as long as "procedures […] include all cryptographic checks included with the specifications in this document. This is important because the checks are essential for the prevention of subtle attacks".

    For example, an implementation may start on one of the permitted curves given in Table 2 of **SP 800-186**, then map to a different curve, perform some operation(s), and map back to the original permitted

curve. This should not impact the CAVP testing which will test the final signature on the original curve.

11. **FIPS 186-5** uses Ed25519 and Ed448 to mean the signature schemes and not the curves themselves, while **SP 800-186** uses Edwards25519 and Edwards448 to mean the curves themselves.
12. Please see IG C.F for additional guidance on this transition as it relates to the RSA key sizes and number of Rabin Miller tests.

C.L SP 800-107 Requirements

Applicable Levels:	All
Original Publishing Date:	July 25, 2023
Effective Date:	July 25, 2023
Last Modified Date:	July 25, 2023
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

Background

NIST has announced plans to withdraw **SP 800-107r1**, *Recommendation for Applications Using Approved Hash Algorithms*. This document provides information on the security properties of approved hash algorithms and additional requirements to ensure the secure use of approved hash algorithms. The information in **SP 800-107r1** is being relocated to the individual standards to which it pertains. In the meantime, this IG will serve to list the requirements from the withdrawn **SP 800-107r1** that modules must still comply with.

Question/Problem

What are the requirements in **SP 800-107r1** that will be relocated to future versions of other documents which are listed below?

Resolution

A. Shall Statements for FIPS 180-4: Secure Hash Standard (SHS) and FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions

1. When truncating a message digest, the following requirements apply where λ is the desired truncated length in bits.
    a. The length of the output block L of the **approved** hash function to be used **shall** be greater than λ, (i.e., $L > \lambda$).
    b. The λ left-most bits of the full-length message digest **shall** be selected as the truncated message digest.
    c. If collision resistance is required in the truncated message digest, λ **shall** be at least twice the required collision resistance strength s, in bits, for the truncated message digest (i.e., $\lambda \geq 2s$).

B. Shall Statements for FIPS 186-5: Digital Signature Standard (DSS)

1. When truncating a message digest for a digital signature, the value of λ **shall** be at least twice the desired security strength in bits required for the digital signature.

C. Shall Statements for FIPS 198-1: The Keyed-Hash Message Authentication Codes (HMAC)

1. An HMAC key **shall** have a security strength that meets or exceeds the security strength required to protect the data over which the HMAC is computed.
2. The HMAC key **shall** be kept secret.
3. HMAC keys **shall** be generated as specified in **SP 800-133**.
4. When an application truncates the HMAC output to generate a *MacTag* to a desired length, λ, the λ left-most bits of the HMAC output **shall** be used as the *MacTag*. The length of λ **shall** be no less than 32 bits.
5. The table in this resolution provides the likelihoods of accepting forged data for different *MacTag* lengths and the maximum allowed number of MAC verification failures for each truncation length using a given value of the HMAC key. This table is intended to assist the

implementers of HMAC applications in security-sensitive systems to assess the security risk associated with using *MacTags*.

Table 1 The probability of forgery versus the number of failed verifications

Length of truncated tag λ in bits	Number of failed verifications allowed (2^t)	Probability of forgery
40	2^{20}	2^{-20}
	2^{30}	2^{-10}
	2^{35}	2^{-5}
64	2^{20}	2^{-44}
	2^{30}	2^{-34}
	2^{35}	2^{-29}
96	2^{20}	2^{-76}
	2^{30}	2^{-66}
	2^{35}	2^{-61}

If the probability of a forgery for a given *MagTag* length and the number of failed *MacTag* verifications is not acceptable for the system, the HMAC key **shall** be changed to a new value before the number of failed MAC verifications reaches 2^t which is presented in the middle column of Table 1.

D. Shall Statements for SP 800-56Ar3, SP 800-56Br2, SP 800-56Cr2, and SP 800-108r1

1. If a derived key is intended to provide s bits of security strength, then each of the following **shall** be equal to or greater than s:

a. The security strength supported by the asymmetric keys, or key-derivation key,

b. The preimage strength of the hash function, or the hash function used in all HMAC constructions used to derive the key,

c. The length of the derived key in bits.

E. Shall Statements for SP 800-90Ar1

1. The hash function used by an RBG **shall** be selected so that the RBG can provide a security strength that meets or exceeds the minimum security strength required for the random bits that it generates.

The CST testing lab **shall** document in TE02.20.01 of the Test Report how the module meets the above requirements.

C.M Legacy Algorithms

Applicable Levels:	All
Original Publishing Date:	July 26, 2024
Effective Date:	July 26, 2024
Last Modified Date:	December 20, 2024
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

Background

SP 800-131Arev2:

The terms “acceptable”, “deprecated”, “**legacy use**” and "disallowed" are used throughout this Recommendation to indicate the approval status of an algorithm. The approval status for an algorithm often will also depend on the length of its key, any domain parameters and the mode or manner in which it is used.

- Acceptable is used to mean that the algorithm and key length in a FIPS or SP is safe to use; no security risk is currently known when used in accordance with any associated guidance. The FIPS 140 Implementation Guideline may indicate additional algorithms that are acceptable for use, but not specified in a FIPS or SP.
- Deprecated means that the algorithm and key length may be used, but the user must accept some security risk. The term is used when discussing the key lengths or algorithms that may be used to apply cryptographic protection.
- Disallowed means that the algorithm or key length is no longer allowed for applying cryptographic protection.
- **Legacy use means that the algorithm or key length may be used only to process already protected information (e.g., to decrypt ciphertext data or to verify a digital signature).**

The use of algorithms and key lengths for which the terms deprecated and **legacy use** are listed require that the user accept some risk [2] that increases over time. If a user determines that the risk is unacceptable, then the algorithm or key length is considered disallowed from the perspective of that user. It is the responsibility of the user or the user’s organization to determine the level of risk that can be tolerated for an application and its associated data and to define any methods for mitigating those risks.

Question/Problem

1. What is the definition of legacy algorithms?
2. What is the current list of legacy algorithms?
3. What are the validation requirements in order for a module to be able to implement a *legacy* algorithm in an approved service?

Resolution

1. Per **SP 800-131Arev2**, “Legacy use means that the algorithm or key length may be used only to process already protected information (e.g., to decrypt ciphertext data or to verify a digital signature).” They are associated with algorithm standards that are no longer approved (i.e., withdrawn). They are *approved* (or *allowed*) only for *legacy* purposes.
2. The table below provides the current list (as of the last modified date of this IG) of the legacy algorithms, their status, applicable standard/IG, and when that algorithm was classified as legacy

[2] For example, if the data was encrypted and transmitted over public networks when the algorithm was still considered secure, it may have been captured (by an adversary) at that time and later decrypted by that adversary when the algorithm was no longer considered secure; thus, the confidentiality of the data would no longer be assured.

based on **SP 800-131A** (all versions), CMVP IGs, and the CMVP Programmatic Transitions webpage:

Algorithm	Status	Standard/IG	Legacy Date
Symmetric Algorithms Used for Decryption / Unwrapping			
Two-key TDEA Unwrapping	Legacy (*allowed*)	IG D.G SP 800-67rev2	2011
Two-key TDEA Decryption	Legacy (*approved*)	SP 800-67rev2	2011
SKIPJACK Decryption / Unwrapping*	Legacy (*approved*)	FIPS 185	2011
Unauthenticated AES Unwrapping	Legacy (*allowed*)	IG D.G	2018
Three-key TDEA Unwrapping	Legacy (*allowed*)	IG D.G SP 800-67rev2	2018 (unauthenticated) 2024
Three-key TDEA Decryption	Legacy (*approved*)	SP 800-67rev2	2024
Algorithms Used for Digital Signature Verification			
ECDSA Signature Verification and Key Verification/Validation: $160 \leq len(n) < 224$ (provides < 112 bits of security strength)*	Legacy (*approved*)	FIPS 186-2	2011
RSA Signature Verification: $1024 \leq len(n) < 2048$ (provides < 112 bits of security strength)	Legacy (*approved*)	FIPS 186 series	2011
SHA-1 within Digital Signature Verification	Legacy (*approved*)	FIPS 180-4 and FIPS 186 series	2011
RSA Signature Verification specific to FIPS 186-2 (e.g., parameters used)	Legacy (*approved*)	FIPS 186-2	September 2020
ECDSA Signature Verification and Key Verification/Validation specific to FIPS 186-2 (e.g., parameters used)*	Legacy (*approved*)	FIPS 186-2	September 2020
ECDSA using curves permitted by IG C.A Category 2 (i.e., not one of the well-known named elliptic curves listed in that IG)*	Legacy (*allowed*)	IGs C.A (Category 2) and C.K (Resolutions 2 and 6)	Feb 5, 2024

DSA Signature Verification and PQG Verification	Legacy (*approved*)	FIPS 186-2*, FIPS 186-4	Feb 5, 2024
RSA/ECDSA Signature Verification specific to FIPS 186-4 (e.g., RSA X9.31)	Legacy (*approved*)	FIPS 186-4	Feb 5, 2024
MAC Verification Algorithms			
HMAC Verification: Key lengths < 112 bits*	Legacy (*approved*)	FIPS 198-1	2011
Two-key TDEA CMAC Verification*	Legacy (*approved*)	SP 800-67rev2	2011
Three-key TDEA CMAC Verification	Legacy (*approved*)	SP 800-67rev2	2024
* Not CAVP testable as of the last Modified Date of this IG			

If an algorithm does not specify the applicable key sizes within this table, it is assumed to apply to all previously approved sizes defined in their respective algorithm standards.

There is currently no scheduled transition for any of these legacy algorithms as these continue to be *approved* (or *allowed*) only for legacy purposes for verifying/decrypting already existing information in case the requirements given in this IG are satisfied.

3. The following **shall** be met for a module to be able to use a legacy algorithm in an approved service:

   a. *Approved* legacy algorithms must be CAVP tested and have a self-test to maintain "*approved*" status. See IG 10.3.A for self-test requirements.

   b. *Allowed* legacy algorithms must be CAVP tested and have a self-test if CAVP testing and self-test requirements exists (see * in the above table for CAVP testing, and IG 10.3.A for self-test requirements).

   c. The module's *approved* CAVP Certs Table must list the *approved* legacy algorithms. The "Legacy" designation is captured in the "Name" field of the SFI table instead of within the CAVP Certs Table. In the case of the *allowed* IG C.A scenario, the "Legacy" designation is captured both in the Algorithm Name column of the Allowed Table, as well as the "Name" field of the SFI table.

   d. The Security Policy Section 2.7 *Algorithm Specific Information* must include the following caveat for the legacy algorithms: "Algorithms designated as "Legacy" can only be used on data that was generated prior to the Legacy Date specified in FIPS 140-3 IG C.M".

   e. If, during module testing, the tester can determine/confirm, with a high degree of confidence, that the data was generated after to the associated algorithm's Legacy Date, then the algorithm must be considered non-approved (as opposed to *approved*/*allowed* for legacy usage), since the data is confirmed to be generated using a non-approved algorithm at the time of generation. This analysis must be included in TE02.20.01. E.g., if a signature verification algorithm is used as an approved integrity technique, and the signing of this code is known to happen after the Legacy Date of that signature verification algorithm per the table above, then this signature verification algorithm cannot be used as an approved integrity technique and can only be considered a non-approved algorithm.

Additional Comments

1. No additional comments.

C.N Requirements for SP 800-208 Schemes

Applicable Levels:	All
Original Publishing Date:	July 26, 2024
Effective Date:	July 26, 2024
Last Modified Date:	September 10, 2024
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01
Relevant Vendor Requirements:	VE02.20.01

Background

Stateful Hash-Based Signatures schemes are specified in SP 800-208. This standard was published in October 2020 and added as an approved security function in SP 800-140Crev2 in May 2022. The standard specifies two stateful hash-based signature (HBS) schemes: the Leighton-Micali Signature (LMS) system and the eXtended Merkel Signature Scheme (XMSS), alongside their multi-tree variants, the Hierarchical Signature System (HSS) and multi-tree XMSS ($XMSS^{MT}$).

As of the publication date of this IG, CAVP tests are undergoing development with testing available for LMS but not yet for XMSS, HSS, or $XMSS^{MT}$.

Alongside the definition of parameters to be used with LMS, XMSS and their multi-tree variants HSS and $XMSS^{MT}$, **SP 800-208** defines requirements for cryptographic modules implementing the algorithms and aimed at managing risks associated with stateful signature schemes based on one-time signatures (OTS).

Cryptographic modules are required by **SP 800-208** to implement key generation when supporting signature generation. Modules may separately support signature verification which can be supported as a standalone service or alongside implementations for key generation and signature generation.

This implementation guidance:

- provides clarifications on several requirements from **SP 800-208**; and
- identifies additional requirements that must be met by FIPS 140-3 certified modules implementing the HBS signature scheme beyond those defined in **SP 800-208**.

Question/Problem

Some ambiguities alongside potential omissions in **SP 800-208** have been publicly documented. Vendors and testing laboratories would like to understand CMVP's interpretation of these ambiguities and omissions including how testing of compliance against **SP 800-208** should be performed. The following questions are addressed by this guidance:

- Do requirements in section 8 of **SP 800-208** apply to modules that support **HBS** signature verification only? (Resolutions 1-2)
- Are there any circumstances where HBS private keys are permitted to cross the cryptographic boundary of the module? (Resolutions 3-4)
- Is key generation for **SP 800-208** permitted to use a CMVP-approved random bit generator seeded from an external source? (Resolution 5)
- How does CMVP interpret **SP 800-208**'s requirement to exclusively support an approved mode with no support for bypass? (Resolution 6)
- Are internal copies of HBS private keys permitted internal to the cryptographic boundary of the module? (Resolution 7)
- Can a module use volatile memory for storage of HBS private keys without storing any component of the private key (including the leaf index) in non-volatile memory? (Resolution 8)
- Can FIPS 140-3 certified cryptographic modules other than those of type: 'hardware module' support

HBS implementations compliant to requirements in Section 8 of **SP 800-208**? (Resolution 9)

- What vendor evidence and test evidence are required to support a claim of conformance to **SP 800-208**? (Resolutions 10-12)

Resolution

Applicability of Requirements

1. Requirements from section 8.1 of **SP 800-208** exclusively apply to modules supporting HBS key generation and signature generation.

   Modules that only support signature verification need to comply with requirements from section 8.2 of **SP 800-208**, but do not need to meet requirements from section 8.1.

Key Import and Export

2. Section 8.1, Key Generation and Signature Generation of **SP 800-208** states:

   "The cryptographic module **shall not** allow for the export of private keying material."

   This resolution confirms that this requirement from **SP 800-208** applies exclusively to private keys used for **SP 800-208** compliant stateful HBS. i.e., This requirement has no impact on a module's ability to export other private key types (such as used with: DH, ECDH, RSA, ECDSA and EdDSA).

3. In addition to the prohibition of the export of private keys, this resolution extends this to include the prohibition of the import of private key material for use with an approved HBS service, including the prohibition of importing private keys as a restore operation from a backup. This is consistent with the security considerations cited in section 9.1 of **SP 800-208** to protect against private key re-use that may occur if a key is separately imported into multiple independent cryptographic modules.

4. Key export as used in **SP 800-208** includes all scenarios (e.g., backup of private keys) where copies of a private key generated by the cryptographic module and intended to be used for HBS cross the module's cryptographic boundary. This includes both in plaintext and ciphertext forms.

Entropy Source

5. Section 8.1 of **SP 800-208** states:

   "The entropy source for any approved random bit generator used in the implementation **shall** be located inside the cryptographic module's physical boundary."

   This resolution interprets "physical boundary" to mean hardware perimeter of a module that was tested to meet the physical security requirements of **ISO/IEC 19790:2012** section 7.7. Note, the hardware perimeter, in the context of this resolution, may be different (smaller) than the defined TOEPP (from IG 9.5.A).

   The module **shall** not use (for **SP 800-208** compliant HBS key generation) an external entropy source that is outside the hardware perimeter to seed the DRBG, even if permitted under IG 9.3.A.

Approved mode of operation and bypass mode

6. Section 8.1 of **SP 800-208** states:

   "a cryptographic module that implements the key generation or signature algorithms **shall** only operate in an approved mode of operation and **shall not** implement a bypass mode."

   This resolution confirms that this statement applies to **SP 800-208** compliant HBS schemes – their keys **shall** be generated using an approved service. Signature generation and signature verification using those keys **shall** use approved services. This statement does not mean that a module cannot be configured to support a non-approved mode of operation. However, **SP 800-208** compliant HBS schemes cannot be non-approved services.

   Cryptographic modules that contain **SP 800-208** compliant HBS schemes in the approved mode may also support non-approved HBS schemes in the non-approved mode (e.g. schemes using hash functions listed by RFC 8554, *Leighton-Micali Hash-Based Signatures* but not permitted by **SP 800-**

208, and hash functions listed by RFC 8391, *XMSS: eXtended Merkle Signature Scheme* but not permitted by **SP 800-208**). For any non-approved service, the module **shall** protect against use of an **SP 800-208** compliant HBS private key used with an approved service with any non-approved service. Note the related requirement is **AS02.22**: *CSPs **shall** [02.22] be exclusive between approved and non-approved services and modes of operation (e.g. not shared or accessed)*.

Cryptographic modules that contain **SP 800-208** compliant HBS schemes may support other non-approved services as permitted by **ISO/IEC 19790:2012**.

Keys used with **SP 800-208** compliant HBS schemes **shall** have all requirements from that standard (and as interpreted or augmented in this IG) enforced for their full life cycle.

This resolution confirms that modules may support a bypass capability as covered in section 7.4.3.2 of **ISO/IEC 19790:2012** and where this is not considered a "bypass mode" as identified in **SP 800-208.**

Bypass mode as identified in **SP 800-208** is interpreted as being any feature that may be supported by a module that allows a module user to temporarily disable any controls technically enforced by the module to meet the requirements of **SP 800-208**. An example control would be the technical enforcement against the export of HBS private keys.

Copying of Keys

7. In addition to the prohibition of the export and import of keys as covered in resolutions 1, 2, 3, and 4 above, this resolution prohibits the creation of independent copies of keys internal to the cryptographic boundary of the module.

   It is common within cryptographic modules to allow the creation of multiple copies of a key that may be assigned to different users of the module. Where this is a stateless key, the use of independent copies of keys by users concurrently accessing the module does not pose any threat to the security of the individual keys. By contrast, if permitted for stateful HBS private keys, if the state is managed independently for each copy of the key internal to the module, this would lead to OTS private key re-use.

Use of Non-Volatile Memory

8. Section 8.1 of **SP 800-208** states:

   "In order to prevent the possible reuse of an OTS private key, when the cryptographic module accepts a request to sign a message, the cryptographic module **shall** increment the leaf index of the private key (q in LMS, idx in XMSS, idx_sig in $XMSS^{MT}$) and **shall** store the incremented leaf index value in nonvolatile storage before exporting a signature value or accepting another request to sign a message."

   The recommendation confirms that the HBS private keys may be stored in either volatile or non-volatile memory. The requirement above only applies where all or part of a HBS private key is stored in non-volatile memory.

Cryptographic Module Type

9. Section 8.1 of **SP 800-208** states:

   "Implementations of the key generation and signature algorithms in this document **shall** only be validated for use within hardware cryptographic modules."

   this is then followed by:

   "The cryptographic modules **shall** be validated to provide FIPS 140-2 or FIPS 140-3 Level 3 or higher physical security, and the operational environment **shall** be non-modifiable or limited."

   Provided the hardware components of any FIPS 140-3 certified module satisfies the latter two requirements, it is not required that modules are module type: 'hardware' and may separately include modules of type 'firmware' and 'hybrid firmware'. For any permitted module type, these **shall** support a certification claim to the Level 3 requirements from section 7.7, Physical security, of **ISO/IEC 19790:2012**.

This resolution excludes software and hybrid software modules on the basis that these are not by definition in **ISO/IEC 19790:2012** permitted to include a limited or non-modifiable operating environment (OE) as required by **SP 800-208**.

Test and Documentation Requirements

10. To assess compliance, the testing laboratory **shall** review vendor evidence of compliance to all requirements from **SP 800-208** including resolution within this IG.
11. The vendor **shall** justify the implementation strategy taken to eliminate the possibility of OTS private key re-use. This **shall** include justification for how the leaf index for HBS private keys is managed to ensure that it is updated and stored in a manner to avoid all situations that could lead to OTS private key re-use. The testing laboratory **shall** review this justification including a review of the algorithm and associated implementation protecting against key reuse, to validate the vendor's claims.
12. The testing laboratory **shall** provide the following information as part of the module's validation report submitted to CMVP and where detailed information **shall** be recorded against TE02.20.01 (may be replaced by a CMVP template to be filled in by the CSTL). All requirements **shall** be validated based on either implementation review or functional testing of the module. Documentation review in isolation is not sufficient to confirm compliance with these requirements.

    For all modules:

    a. The tester **shall** confirm whether the module performs HBS key generation, signature generation, and/or signature verification.
    b. The tester **shall** confirm whether the module supports any non-approved HBS algorithms in addition to its **SP 800-208** compliant HBS implementation.
    c. The tester **shall** confirm which LMS, LM-OTS, XMSS and XMSS^{MT} parameter sets the module supports.
    d. The tester **shall** confirm that they verified that the combinations of LMS and LM-OTS parameter set adhere to **SP 800-208**, Section 4 including checking $n = m$, the hashing algorithm selected is the same for LM-OTS compared to LMS and height of the supported tree is enforced to be one of the following: 5, 10, 15, 20 or 25.
    e. The tester **shall** confirm that they verified that the combinations of XMSS and XMSS^{MT} parameter sets adhere to **SP 800-208**, section 5.

    For modules supporting key generation and signature generation:

    f. The tester **shall** document how HBS private key state is updated and recorded prior to completing a new signature.
    g. The tester **shall** confirm that they verified the module meets level 3 or higher physical security.
    h. The tester **shall** confirm that they verified the operational environment is non-modifiable or limited.
    i. If the module supports non-approved HBS, the tester **shall** confirm how they verified that non-approved services are prevented from accessing **SP 800-208** HBS private keys and state (related to **AS02.22**).
    j. If the module supports non-approved HBS, the tester **shall** confirm they verified the approved **SP 800-208** HBS services are prevented from accessing non-approved HBS keys and state (related to **AS02.22**).
    k. the tester **shall** confirm how they verified the module does not permit any export or import of HBS private keys or their state.
    l. the tester **shall** confirm they verified the entropy source used by the module to meet **SP 800-208** complies with Resolution 5 of this IG and **SP 800-90B** requirements.

m. the tester **shall** confirm they verified the module prevents the use of an OTS key to generate a digital signature more than one time.

n. the tester **shall** confirm the module cannot create an independent copy of an HBS key.

o. The tester **shall** confirm whether the module supports non-volatile or volatile storage of HBS private keys.

p. If the module supports non-volatile storage of HBS private keys, the tester **shall** confirm that the state information for the key (at minimum the leaf index value) is updated in the non-volatile memory before exporting a signature value or accepting a request to sign another message.

q. The tester **shall** confirm the module does not support a "bypass mode" as defined in Resolution 6.

r. Per 2.6.2 of the FIPS 140-3 Management Manual, the tester **shall** confirm that the module is compliant with all applicable requirements of **SP 800-208** where 'shall' statements are not addressed by CAVP testing.

Additional Comments

1. Requirements for CAST for LMS, HSS, XMSS and XMSS^{TM} are defined in IG 10.3.A.
2. This IG focuses exclusively on the current active version of **SP 800-208**. Should that standard be updated in the future (or allowances created by the NIST Cryptographic Technology Group (CTG) be published), resolutions within this IG will be reviewed and as necessary updated at that time.
3. HBS private key as referred to in this IG refers to any combination of:
   a. the random seeds, SEED (LMS) and S_XMSS (XMSS); and (optionally)
   b. the pseudo-randomly generated Leighton-Micali One-Time Signature (LM-OTS) private keys (LMS and HSS) and Winternitz One-Time Signature Plus (WOTS^{+}) private keys (XMSS and XMSS^{TM}).
4. **SP 800-208** is exclusively approved for use with FIPS 140-3 validated modules and has never been approved for FIPS 140-2 modules despite references to FIPS 140-2 by **SP 800-208**.
5. See IG C.O for the requirements for vendor affirming HSS. Besides HSS, there is currently no way to vendor affirm algorithms in **SP 800-208**. A vendor **shall** rely on the CAVP tests to claim compliance to **SP 800-208**.

C.O Requirements for SP 800-208 HSS Vendor Affirmation

Applicable Levels:	All
Original Publishing Date:	September 10, 2024
Effective Date:	September 10, 2024
Last Modified Date:	September 10, 2024
Relevant Assertions:	AS02.20, AS10.27
Relevant Test Requirements:	TE02.20.03, TE02.20.04, TE.10.27.01
Relevant Vendor Requirements:	VE02.20.03, VE02.20.04, VE10.27.01, VE10.27.02, VE10.27.03

Background

In October 2020, NIST released **SP 800-208**, the Recommendation for Stateful Hash-Based Signature Schemes. This publication specifies the Leighton-Micali Signature (LMS) scheme and eXtended Merkle Signature Scheme (XMSS), along with their multi-tree variants, HSS and XMSS^{MT}. These signature schemes were included as approved signature schemes in **SP 800-140Crev1** in May 2022, and their self-test requirements were specified in FIPS 140-3 IG 10.3.A in March 2023. Support for the LMS scheme was added to the ACVTS production server in April 2023, which cleared the way for LMS to be used in an approved mode of operation. However, the other schemes (including HSS) remain untestable by the CAVP as of the last modified date of this IG.

Question/Problem

Can an HSS implementation be vendor-affirmed, provided all applicable requirements (e.g. CASTs) are met, the underlying LMS operations are CAVP tested, and the correct implementation of HSS is verified by the CST testing lab?

Resolution

Vendor affirmation is available for HSS implementations, provided the following conditions are met:

1. The required HSS cryptographic algorithm self-tests, as specified in IG 10.3.A, **shall** be performed.
2. If HSS key generation is implemented, the underlying LMS key generation and LMS signature generation operations **shall** have CAVP certificates.
3. If HSS signature generation is implemented, the underlying LMS key generation and LMS signature generation operations **shall** have CAVP certificates.
4. If HSS signature verification is implemented, the underlying LMS signature verification operation **shall** have a CAVP certificate.
5. **SP 800-208** explicitly allows different LMS levels within the HSS tree to use different parameter sets. All parameter sets implemented by the cryptographic module within the HSS tree **shall** have CAVP certificates as specified above.
6. The CSTL **shall** verify the correct implementation of each supported HSS operation according to RFC 8554 Section 6.1 (Key Generation), Section 6.2 (Signature Generation), and Section 6.3 (Signature Verification) through source code review. The CSTL **shall** document the results of the review, and include a reference to this IG, in TE02.20.04 of the Test Report.
7. HSS **shall** be listed in the “Vendor-Affirmed Algorithms” table of the Security Policy. The Security Policy **shall** state that the LMS operations used by the HSS implementation were tested in accordance with this IG. This statement **shall** reference the associated CAVP certificates and the LMS algorithm **shall** be listed in the “Approved Algorithms” table.

Additional Comments

1. This IG does not remove or modify any requirements specified in Section 8 of **SP 800-208**. The CST testing lab **shall** verify the compliance of the HSS implementation to Section 8, as required for all stateful hash-based signature algorithms. For additional guidance, see IG C.N *Requirements for SP 800-208 schemes*.
2. When CAVP testing for HSS is released on the ACVTS production server, the transition process specified in the FIPS 140-3 Management Manual Section 7.2.2 *Transitioning from vendor affirmed to CAVP Testing* **shall** be followed.

Annex D – Approved sensitive security parameter generation and establishment methods

D.A Acceptable SSP Establishment Protocols

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

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 19790:2012** 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 and document the strength of the established key.

D.B Strength of SSP Establishment Methods

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	April 18, 2025
Relevant Assertions:	AS09.10, AS09.11
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

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 2, which provides comparable security strengths for the approved algorithms.

Table 2: 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	$L = 2048$ $N = 224$	$k = 2048$	$f = 224\text{-}255$
128	AES-128	$L = 3072$ $N = 256$	$k = 3072$	$f = 256\text{-}383$
192	AES-192	$L = 7680$ $N = 384$	$k = 7680$	$f = 384\text{-}511$
256	AES-256	$L = 15{,}360$ $N = 512$	$k = 15{,}360$	$f = 512+$

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.

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** and **FIPS 186-5** 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 **FIPS 186-5**, 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:

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

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:

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

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 the applicable caveat per Table 10 (SFI) in MIS Table Descriptions.

Additional Comments

As of the publishing of this document, **SP 800-57** ***Part 1***, is not addressed in **SP 800-131Arev2**, 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 4 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.
    a. "Acceptable" indicates that the algorithm or key length is currently considered to be secure.
    b. "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).
    c. "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 4: 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	October 21, 2024
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**? Is the AES-CBC-MAC within OTAR permitted in FIPS 140-3?

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.

For a period of two years as of 11/1/2021, the CMVP will allow AES-CBC-MAC within OTAR for FIPS 140-3. AES-CBC-MAC **shall** appear on the non-approved but allowed algorithms list for the module submission. There will be no CAVP tests for AES-CBC-MAC within OTAR as it is not an approved algorithm.

The Telecommunications Industry Association (TIA) TR8.3 Encryption Subcommittee has updated the OTAR specification to include the CMAC algorithm. The published revision (dated August 2, 2023) is TIA-102.AACA-C, Project 25 – Digital Radio Over-The-Air-Rekeying (OTAR) Messages and Procedures Standard.

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**.
2. This IG also applies to the TLS 1.3 KDF CVL.

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

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

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**, **FIPS 186-5**, **SP 800-186**, **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 SP 800-186** (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** and **FIPS 186-5** are digital signature standards, some of their 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 800-56Arev3.

D.E Moved to W.1

D.F Key Agreement Methods

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	July 26, 2024
Relevant Assertions:	AS09.10
Relevant Test Requirements:	TE09.10.01-02
Relevant Vendor Requirements:	VE09.10.01

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 800-56Brev2**-compliant RSA-based key agreement schemes. The module's implementation of a key agreement scheme **shall** show compliance with **SP 800-56Brev2**.

The implementations claiming compliance to **SP 800-56Brev2** 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 800-56Brev2**. This compliance will be annotated as **KAS-IFC-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 800-56Brev2**. 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 800-56Brev2**.

When path (2) is chosen, the CAVP testing may be performed either end-to-end, in which case the vendor is issued a **KAS-IFC** algorithm certificate [1], 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

[1] A KAS-IFC certificate will only be issued if a key-agreement scheme implements a key derivation function from SP 800-56Crev1 or rev2.

confirmation step in Figure 7, 9, 10, or 11 in **SP 800-56Brev2**. The key derivation function **shall** comply to either **SP 800-56Crev1** 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 with 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 applicable self-tests

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 800-56Brev2**, 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 800-56Brev2**-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 800-56Brev2** 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 800-56Brev2** 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 800-56Brev2**-compliant key agreement scheme. All key derivation functions not included in the **SP 800-56Brev2** CASTs **shall** have their own self-tests prior to their first operational use.

Scenario 2. Approved **SP 800-56Arev3**-compliant Discrete Logarithm Cryptographic (DLC)-based key agreement scheme. The module's implementation of a key agreement scheme **shall** show compliance with **SP 800-56Arev3**. The implementations claiming compliance to this scenario 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 or more of the key agreement schemes in Section 6 of **SP 800-56Arev3**. This compliance will be annotated as **KAS-ECC-SSC** or **KAS-FFC-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 **800-56Arev3**. This may optionally be followed by the unilateral or bilateral key confirmation shown in Section 5.9 of SP **800-56Arev3**.

When path (2) is chosen, the CAVP testing may be performed either end-to-end, in which case the vendor is issued a **KAS-ECC** or **KAS-FFC** certificate [2], 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 800-56Arev3**. The key derivation function **shall** comply to either SP **800-56Crev1** 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.

[2] A KAS-ECC or KAS-FFC certificate demonstrating compliance with Scenario 2 will only be issued if a key-agreement scheme implements a key derivation function from SP 800-56Crev1 or rev2.

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 800-56Arev3**.

The applicable self-tests

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 800-56Arev3**: 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 800-56Arev3**, **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 800-56Arev3**.

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 800-56Arev3** 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 800-56Arev3** 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 800-56Arev3**). 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 800-56Arev3** CASTs **shall** have their own self-test prior to their first operational use (see Additional Comment 11 below and IG 10.3.A Resolution 11 for additional self-test guidance on KDFs used within a KAS).

Allowed methods for key agreement.

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 800-56Arev3** 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 annotation details for the key agreement schemes (KAS) can be found in Table 10 (SFI) in MIS Table Descriptions.
5. For the appropriate assurances as required in Section 5.6.2 of **SP 800-56Arev3** and Section 6.4 of **SP 800-56Brev2**, the module **shall** on its own obtain these assurances, unless it is impossible to do so (e.g., the module is a software library providing RSA key encapsulation or un-encapsulation and not both, and therefore cannot meet the assurances that require access to both keys) in which case the module's Security Policy **shall** provide the guidance on how these assurances can be obtained. The module **shall** obtain these assurances if it receives the necessary input(s) to perform them.
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 Security Functions Implementation (SFI) table **shall** include the **KAS** entry and reference the applicable tested components within these entries (e.g., KAS FFC SSC + KDA + KAS KC). The Approved Algorithm list **shall** also include 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 800-56Crev1** or **rev2** or in IG 2.4.B. Note that all **SP 800-135rev1** KDFs and the TLS 1.3 KDF are included in IG 2.4.B.
10. The acronym **SSC** stands for "shared secret computation".
11. Self-tests are required for the CVL KDFs used as part of an approved key agreement scheme specified in Scenario 1 path (2) or Scenario 2 path (2). CVL KDFs that are not used as part of an approved key agreement scheme do not require their own self-tests. The KDA self-tests are always required to meet IG 10.3.A regardless of whether or not they are used within a scenario specified in this IG.
12. When implementing a key agreement scheme (or a shared secret computation as part of a key agreement scheme), the module's Security Policy **shall** indicate whether the scheme is of the Diffie Hellman or the MQV variety.

D.G Key Transport Methods

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	July 25, 2023
Transition End Dates	**December 31, 2023 – See Below**
Relevant Assertions:	AS09.10
Relevant Test Requirements:	TE09.10.01, TE09.10.02
Relevant Vendor Requirements:	VE09.10.01

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 [3]. 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-IFC entry **shall** be added to the module's validation certificate as shown in MIS Table Descriptions (see "CAVP Algorithms" and Table 10 SFI). The Security Policy **shall** document the tested RSA modulus sizes, the method (from **SP 800-56Brev2** Section 6.3) of RSA key generation, the tested key confirmation (if applicable) 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

[3] The state existing between two entities when they share at least one cryptographic SSP.

the module's validation certificate **shall** include an RSA key generation 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 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 as shown in MIS Table Descriptions (see "CAVP Algorithms" and Table 10 SFI).

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 for *legacy* use. Key wrapping is not allowed if the algorithm does not meet the requirements of **SP 800-38F**.

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 $M^e (mod\ N)$, while the decryption CAST consists of checking the value of $M^d (mod\ N)$. 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. Approved key transport methods **shall** be annotated per MIS Table Descriptions: *Approved Algorithms*, Table 10 *SFI* (as a "KTS").
5. Allowed key transport methods **shall** be annotated per MIS Table Descriptions: *Approved Algorithms*, Table 7 *Non-Approved, Allowed Algorithms*, Table 10 *SFI* (as a "KTS").
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 was only available for a short time period (a few months) before the transition went into effect at the end of 2020, so there was little value in permitting submissions that would become disallowed come January 2021.

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

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	March 17, 2023
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01-2
Relevant Vendor Requirements:	VE02.20.01

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 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 800-133rev2** 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, …, 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, …, K_n are generated from a password.

The module's test report **shall** explain how the keys K_1, …, K_n are generated, how the values D_1, …, 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 800-133rev2** corresponds to Section 6.6 of **SP 800-133rev1**.
4. The method of generating a key by a key-extraction process defined in item 3 of Section 6.3 of **SP 800-133rev2** 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.
5. There is no specific CAST requirement if vendor affirmation is claimed for **SP 800-133**, other than the CASTs required for the underlying approved algorithms, per IG 10.3.A.

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

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: $R_N \rightarrow \{0,1, \ldots, k\text{-}1\}$ be a function on N-bit strings with integer output in the range 1 to k, where k is an arbitrary positive integer. Let $\{P_1, P_2, \ldots, P_k\}$ 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:

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

   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: $P_i(r_2) = (r_2$ 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, '10' corresponds to 2, '11' 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 XOR $A_{F(r_1)}$. Note that in this example, $2 \le N \le M$, 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, $P_i(r_2)$ = 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, $P_i(r_2)$ = 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 $F(r_1)$ = SHA256(r_1) mod 10. Then the post-processed output $P_{F(r_1)}(r_2)$ would be AES($key_{\text{SHA256}(r1) \bmod 10}$, r_2). Note that in this case, $4 \le N \le M$, 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 = SHA256(128-bit representation of i). Let $F(r_1)$ = SHA256(r_1). The output $P_{F(r_1)}(r_2)$ of the post-processing would be AES(SHA256(r_1), r_2). Note that is 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:

   $$P_i(B_1||B_2|| \ldots||B_{M/8}) = SBOX_i(B_1)||SBOX_i(B_2)||\ldots||SBOX_i(B_{M/8})$$

   For specificity, suppose that $M = 128$; there are just 2 byte permutations to choose from, SBOX$_0$ and SBOX$_1$; 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 $r_2 = B_1||B_2|| \ldots||B_{16}$ would be $\text{SBOX}_{\text{parity}(r1)}(B_1) \,||\, \text{SBOX}_{\text{parity}(r1)}(B_2) \,||\ldots||\, \text{SBOX}_{\text{parity}(r1)}(B_{16})$. To complete the example, suppose that the two byte permutations are specified as:

- SBOX_0 = the AES SBOX, and
- SBOX_1 = 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	November 5, 2021
Relevant Assertions:	AS09.08, AS09.09
Relevant Test Requirements:	TE09.08.01, TE09.08.01 TE09.09.01, TE09.09.02
Relevant Vendor Requirements:	VE09.08.01, VE09.08.01 VE09.09.01, VE09.09.02

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 an ESV or one of the following entries on the approved algorithm line: **ENT (P)** or **ENT (NP)** where P stands for physical and NP for non-physical. See IG D.O for the precise definition of each entry.

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 3.2.2 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 legacy entries, ENT (P) and ENT (NP), on the approved algorithm line do not include the algorithm certificate numbers. For new submissions, ESV certification is required.
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**.
10. The CMVP allows other parties, such as the vendor or a third-party consultant, to contribute to or to write the labs' entropy source description and its heuristic entropy analysis if the following conditions are met:

    a. The lab **shall** be responsible for the content submitted. The lab **shall** perform a thorough review of the submission, be accountable for the content of the Entropy Test report and respond to any CMVP comments. As a result, if the report contains serious shortcomings

the CMVP will hold the lab responsible (e.g., by assigning an ECR in accordance with the CMVP FIPS 140-3 Management Manual).

b. The lab **shall** communicate with the CMVP. The CMVP will communicate with the lab and not the vendor or a third-party since it is the lab that the CMVP and the NVLAP have accredited to perform an independent review and testing. The lab **shall not** act solely as an intermediary by passing comments from the CMVP to the vendor or a third party as the lab **shall** possess a full understanding of the content of the Entropy Test report. In some rare cases, the vendor or a third-party may be invited to a meeting between the CMVP and the lab.

c. The lab **shall** explain how the collected entropy is used by the module. Specifics regarding how the module makes use of the entropy generated **shall** be the responsibility of the lab (e.g., the full entropy requirement for a CTR_DRBG without a derivation function or internal entropy generation to justify the properties of various algorithm parameters, including IVs).

D.K Interpretation of SP 800-90B Requirements

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	March 17, 2023
Relevant Assertions:	AS02.20, AS09.08, AS09.09
Relevant Test Requirements:	TE02.20.01, TE09.08.01, TE09.08.02, TE09.09.01, TE09.09.02
Relevant Vendor Requirements:	VE02.20.01, VE09.08.01, VE09.08.02, VE09.09.01, VE09.09.02

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

Digitization

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 that 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 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).

IID Claim

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 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).

Conditioning Components

3. For Section 3.1.5, any 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 provided. 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 conditioning [4], 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 as having "full entropy", so the output of any entropy source whose conditioning chain ends with a non-vetted non-bijective conditioning component cannot be considered as having "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] 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.

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 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 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}).

    There are additional considerations when the stateful conditioning component is a DRBG. For every request to GetEntropy from the entropy source, at least n_{in} bits must be provided to the stateful conditioning component (where n_{in} is defined for the conditioning component and provides h_{in} bits of entropy). If the conditioning component is a DRBG, the conditioning component **shall** have prediction resistance enabled, or **shall** be reseeded before requesting an output using at least n_{in} bits from the noise source or the previous conditioning component. A stateful conditioning component **shall not** output any bits without obtaining at least n_{in} bits from the primary noise source or the previous conditioning component. A DRBG is not a bijective conditioning function. The length of the output of the DRBG **shall** be at most equal to the security strength of the DRBG. If the DRBG is used in other contexts than as a conditioning component for the entropy source, the DRBG **shall** be reseeded both before and after bits are requested in the other context. This may occur when one DRBG instance has been instantiated to serve multiple applications such as within an operating system. The submitter **shall** describe how the DRBG is initialized, the security strength, and if applicable, how the DRBG is rekeyed.

Note 1. 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.

Note 2. The maximum number of bits per request from a DRBG used as a conditioning component stated in this resolution is smaller than the specified max_number_of_bits_per_request provided in **SP 800-90A**. The maximum set in this resolution takes precedence only in this use case of a DRBG.

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, for a conditioning function to qualify as "vetted", it **shall** consist solely of one of the listed vetted functions in this section or this IG. 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).

   Note 1. A CAVP-validated **SP 800-90A** DRBG may be considered a vetted conditioning component. A CTR-DRBG, to be considered vetted, **shall** utilize the derivation function, or **shall** be provided with at least $seedlen$ bits of entropy as input, per request to GenerateBits where $seedlen$ is defined in Table 3 from **SP 800-90A**. The DRBG **shall** be seeded with h_{in} greater than or equal to the claimed security strength of the DRBG, in accordance with **SP 800-90A**. The DRBG **shall** also meet the criteria mentioned in Resolution #5. DRBGs that do not meet the listed criteria in this note may be submitted as non-vetted conditioning components.

8. For Section 3.1.5 and its subsections, nw **shall not** 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 the 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 $h_{\text{out}} = h_{\text{in}}$. 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 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.

Primary and Additional Noise Sources

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 $H_{\text{submitter}}$ 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 $H_{\text{submitter}}$ 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 $H_{\text{submitter}}$ value **shall not** be the result of some general statistical testing process that does not account for the design of the noise source.

Health Tests

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, $H_{\text{submitter}}$ and $H = \min(H_r, H_c, H_I)$ 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).

Full Entropy

19. To receive full entropy from the output of a conditioning component, the following criteria must be met:
    - The conditioning component **shall** be vetted,
    - h_{in} **shall** be greater than or equal to $n_{\text{out}} + 64$ bits,
    - n_{out} **shall** be less than or equal to the security strength of the cryptographic function used as the conditioning component.

    Note 1. If n_{in} bits of full entropy are provided to a vetted conditioning component, then the output of the conditioning component will maintain full entropy.

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 CMVP has published an implementation of these tests [5] 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 Entropy Assessment Report submitted by the lab **shall** describe 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.

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

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 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 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	November 5, 2021
Relevant Assertions:	AS09.01, AS09.04, AS09.05, AS09.08, AS09.25, AS09.26
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

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.

   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.

2. In the case of the CTR_DRBG, the test report **shall** indicate if a derivation function is used during the instantiation and reseeding. If the derivation function is not used, the test report **shall** demonstrate that the DRBG is seeded by an entropy source producing the full-entropy outputs, as required in Section 10.2.1 of **SP 800-90A**. An entropy source seeding the CTR_DRBG without a derivation

function **shall** be located inside the module's physical perimeter and provide full entropy as evaluated by the CMVP.

3. **SP 800-90A** Section 10.2.1 references "an approved RBG" as another option to seed the CTR_DRBG, but this IG only allows an entropy source producing full-entropy outputs. The reason for this is the "approved RBG" is referring to the future RBG construction – specifically, RBG3 - that is shown (as of the latest modification date of this IG) in the draft of **SP 800-90C**.

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

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

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 800-133rev1**, "Symmetric Keys Derived from a Pre-shared Key." **SP 800-108** KDFs may not be used to generate asymmetric keys directly.

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 800-135rev1**. 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:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	September 21, 2020
Relevant Assertions:	AS09.09
Relevant Test Requirements:	TE09.09.01 and TE09.09.02
Relevant Vendor Requirements:	VE09.09.01 and VE09.09.02

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.

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.

D.O Combining Entropy from Multiple Sources

Applicable Levels:	All
Original Publishing Date:	May 4, 2021
Effective Date:	May 4, 2021
Last Modified Date:	May 4, 2021
Relevant Assertions:	AS02.20, AS09.08, AS09.09
Relevant Test Requirements:	TE02.20.01, TE09.08.01, TE09.08.02, TE09.09.01, TE09.09.02
Relevant Vendor Requirements:	VE02.20.01, VE09.08.01, VE09.08.02, VE09.09.01, VE09.09.02

Background

SP 800-90B, published in January 2018, specifies that entropy from only one *noise* source may be credited when estimating the output entropy of an *entropy* source. See Figure 1 and Section 3.1.6 in **SP 800-90B**. Multiple copies of the same physical noise source are viewed as a single noise source. An output from different noise sources may be concatenated together before invoking the conditional component(s) of an entropy source, but only one, primary, noise source produces entropy creditable towards the estimated entropy yield of the source. Again, see Section 3.1.6 of **SP 800-90B** for details.

The scope of **SP 800-90B** is limited to showing the requirements for a single entropy source. It does not say anything about combining entropy from multiple sources. This will be addressed in **SP 800-90C** when this standard, now in the draft form, gets published.

Question/Problem

Prior to the publication of **SP 800-90C**, can a cryptographic module combine entropy from various sources? If so, how to estimate the total amount of entropy generated by these sources?

Resolution

Entropy outputs from the multiple independent **SP 800-90B**-compliant entropy sources may be concatenated together to provide a DRBG seed.

Each entropy source is identified as either physical or non-physical. The definitions of the physical and non-physical noise sources are given in Section 2.2.1 of **SP 800-90B**. An entropy source is called physical or non-physical following the classification of the entropy-providing noise source within the entropy source.

Section 3.3 of the current (January 2021) draft of **SP 800-90C** offers two methods for counting entropy collected from multiple entropy sources. Each method allows concatenating the outputs of any number of both physical and non-physical sources. Method 1 counts only the entropy in the physical sources while Method 2 allows counting the entropy from both physical and non-physical sources. A valid min entropy claim for the concatenation of the output of multiple entropy sources is a sum of the individual entropies produced by a subset of those sources whose credited random behavior is independent of all other random behavior credited as providing entropy in that concatenation. If the vendor demonstrates the independence of all entropy sources contributing to the concatenated bitstring then the total entropy is the sum of the individual entropies produced by the sources. If sources S1, S2 and S3 contribute entropy to the module, with S1 and S3 being the physical sources and S2 being non-physical, and the amounts of entropy generated by the sources are, correspondently, E1, E2 and E3, then Method 1 estimates the total entropy as E1 + E3, while the Method 2's estimate is E1 + E2 + E3.

While it appears that Method 2 is more advantageous as it produces, in the presence of non-physical sources, a higher entropy estimate, the vendor and the user **shall** be aware that, as pointed out in the draft of **SP 800-90C**, the entropy produced by a validated physical source is generally more reliable than the entropy produced by a validated non-physical source. Certain constructions described in the draft of **SP 800-90C** will only count

entropy using Method 1. See, for example, the rules in Section 5 of that standard's draft for an RBG1 construction instantiation.

Prior to the publication of **SP 800-90C**, the CMVP will accept both Method 1 and Method 2 of entropy estimation.

It has become necessary to distinguish between the all-physical and not-all-physical validated entropy sources in the modules' certificates. The differentiation is specified using either the ESV certificate, or ENT (P) and ENT (NP) entries for the validated entropy sources. In line with the logic and the requirements of the current draft of **SP 800-56C**, the ENT (P) notation in the certificate indicates that all validated entropy sources creditable toward the module's total entropy estimate are physical; ENT (NP), that at least some of the creditable sources are non-physical. The Security Policy **shall** further explain the nature of the module's entropy sources, specify which of them are creditable, and indicate if Method 1 or Method 2 is used for entropy calculation.

Additional Comments

1. If the multiple entropy sources contributing to the concatenated string are mutually dependent then at most one of these mutually dependent sources may be credited; in this instance, the vendor may select which entropy source should be credited.
2. The current draft of **SP 800-90C** (Section 4.3) allows external conditioning of entropy sources: the conditioning that is performed outside an entropy source. An external conditioning might be useful when it is necessary to obtain a full entropy bitstring. At this time, the Implementation Guidance does not allow this form of conditioning.
3. When full entropy is required, such as when generating a seed for a CTR_DRBG without a derivation function, each of the contributing entropy sources **shall** be shown to generate the full-entropy bitstrings.
4. Entropy sources can be combined only by having their output strings concatenated, as shown in the draft of **SP 800-90C**. This holds true even if the vendor credits the entropy from only one of these sources.
5. In Section 3.5 of the current draft of **SP 800-90C** (January 2021) an entropy source is considered independent of another entropy source if their entropy source security boundaries do not overlap. This is a sufficient but not a necessary condition for an independence. While a true independence of entropy sources located within the same entropy source security boundary is difficult to achieve, if the sources are very different in nature (e.g., one is physical while another one is driven by certain software actions) then it may be possible to make a natural convincing heuristic argument for independence. In all cases, the vendor's arguments supporting their independence will be considered by the CMVP. More on the RBG security boundaries can be found in Section 3.6 of the draft of **SP 800-90C**.

D.P SP 800-56Crev2 One-Step Key Derivation Function Without a Counter

Applicable Levels:	All
Original Publishing Date:	November 5, 2021
Effective Date:	November 5, 2021
Last Modified Date:	November 5, 2021
Relevant Assertions:	AS09.09
Relevant Test Requirements:	TE09.09.01 and TE09.09.02
Relevant Vendor Requirements:	VE09.09.01 and VE09.09.02

Background

NIST Special Publication **800-56Crev2** Section 4 outlines a one-step key derivation function for use in an approved key establishment scheme. The key derivation function uses a counter to ensure uniqueness while iterating through a loop applying the auxiliary function H to the concatenation of the shared secret Z and the FixedInfo field. This allows the KDF to derive keys longer than the output of the auxiliary function used.

Question/Problem

Is it allowed to drop the counter inclusion in the **SP 800-56Crev2** one-step key derivation function when the implementation restricts the derived key to be no longer than the output of the auxiliary function used?

Resolution

A new approved algorithm definition is provided for this case. This is the no-iteration/no-counter variation of one-step key derivation. Implementations using this algorithm description **shall** proceed as follows:

Function call: KDM(*Z*, *OtherInput*).

Options for the Auxiliary Function H:

The options remain the same from those outlined in Section 4.1 in **SP 800-56Crev2**.

Implementation-Dependent Parameters:

The implementation-dependent parameters remain the same from those outlined in Section 4.1 in **SP 800-56Crev2**.

Input:

All the input requirements remain the same from those outlined in Section 4.1 in **SP 800-56Crev2** outside of one change. The only change to the input parameter requirements is in item 2b in Section 4.1 in **SP 800-56Crev2**. To meet the requirements of this IG, use the following definition of *L*.

L – A positive integer that indicates the length (in bits) of the secret keying material to be derived; *L* **shall not** exceed *H_outputBits*.

Process:

1. If $L > H\_outputBits$ or $L \leq 0$, output an error indicator and exit this process without performing the remaining actions (i.e., omit steps 2 through 4).
2. If $Z \parallel FixedInfo$ is more than *max_H_inputBits* bits long, then output an error indicator and exit this process without performing any of the remaining actions (i.e., omit steps 3 and 4).
3. Set *DerivedKeyingMaterial* equal to the leftmost *L* bits of $H(Z \parallel FixedInfo)$.
4. Output *DerivedKeyingMaterial*.

Output:

The bit string *DerivedKeyingMaterial* of length *L* bits (or an error indicator).

Notes:

The notes remain the same from those outlined in Section 4.1 in **SP 800-56Crev2**.

Additional Comments

1. This solution was provided by the Cryptographic Technologies Group at NIST. It may appear in a future revision of **SP 800-56C**.
2. CAVP testing of the updated algorithm description is available under the algorithm name: “KDA”, mode: “OneStepNoCounter”, revision: “Sp800-56Cr2”.
3. The self-test requirement for the algorithm specified in this IG is the same as those under the KDA bullet in IG 10.3.A for the one-step KDF. If the module supports multiple one-step KDAs – one compliant to this IG and the other(s) to **SP 800-56C** – then only one self-test is required if the underlying implementation is the same. The Security Policy **shall** explain how each KDA is used by the module.

D.Q Transition of the TLS 1.2 KDF to Support the Extended Master Secret

Applicable Levels:	All
Original Publishing Date:	May 16, 2022
Effective Date:	May 16, 2022
Last Modified Date:	March 17, 2023
Transition End Dates	**May 16, 2023 – TLS 1.2** **1 year after CAVP testing becomes available – TLS 1.0/1.1**
Relevant Assertions:	AS02.20
Relevant Test Requirements:	TE02.20.01-2
Relevant Vendor Requirements:	VE02.20.01

Background

The TLS 1.2 key derivation function (KDF) has been included, per IG 2.4.B, in the Component Validation List (CVL). These components of the approved algorithms are testable by the CAVP and can be used in the approved mode. IG 2.4.B references **SP 800-135rev1** as the standard where many of the key derivation functions are defined. For the TLS 1.2 KDF, the **SP 800-135rev1** standard points, in turn, to the definition of this key derivation function in the IETF RFC 5246 published in August 2008.

In September 2015 the IETF published RFC 7627 which updated the definition of the TLS 1.2 KDF. Specifically, the computation of the master_secret parameter shown in Section 4 of RFC 7627 is different from the way the master_secret is computed in Section 8.1 of RFC 5246. The master_secret defined in RFC 7627 is often called the extended master secret. **SP 800-52rev2**, published in August 2019, says, in Section 3.4.1.3, that the extended master secret shall be used. The CAVP, which has been previously testing the TLS 1.2 KDF implementations per RFC 5246, has introduced a test for the TLS 1.2 KDF with the extended master secret.

Question/Problem

(1) Can an already-validated module continue using the TLS 1.2 KDF defined in RFC 5246?

(2) Which version of the TLS 1.2 KDF shall be used in the new submissions to the CMVP?

Resolution

The already-validated modules and those with Test Reports submitted to the CMVP no later than one year after the publication date of this IG can remain unchanged.

A new validation, or any revalidation that extends the module's sunset date, submitted more than **one year after the publication date of this IG** **shall** use the extended master secret in the TLS 1.2 KDF. The revalidations not extending the module's sunset date are not subject to this requirement.

The CAVP will support testing both the RFC 5246 and RFC 7627 TLS 1.2 KDFs until **one year after the publication date of this IG** at which time they will remove testing for the RFC 5246-compliant solutions. This testing may still be granted but only on a case-by-case basis (e.g., adding a new OE with no change to the sunset date). The CVL algorithm validation certificate will indicate if the tested TLS 1.2 KDF was implemented with the extended master secret.

The module's Security Policy **shall** identify which version(s) of the TLS 1.2 KDF (RFC 5246 and/or RFC 7627) the module supports. If a version of the TLS 1.2 KDF is not tested by the CAVP then it **shall not** be used in the approved mode.

Additional Comments

1. The TLS 1.0 and TLS 1.1 KDFs can also use either the master secret or the extended master secret as input into the HMAC-MD5/HMAC-SHA-1 PRF described in Section 4.2.1 of **SP 800-135rev1**. However, it can only be used in the approved mode if CAVP tested. If no CAVP testing is available, there will be no vendor affirmed option. There will be a transition of **one (1) year after the**

availability of this testing. Besides the date, the same transition rules apply to the use of the master secret in these KDFs as what is stated in this IG for the TLS 1.2 KDF.

2. This IG does not affect the status of and the testing procedure for the TLS 1.3 KDF.
3. The definition of the "original" master secret is given in Section 8.1 of RFC 5246. The extended master secret parameter is defined in Section 4 of RFC 7627. A comparison of these two definitions shows that while the original master secret is based on the pre_master_secret and the ClientHello.random and ServerHello.random fields defined in RFC 5246, the extended master secret is based on the pre_master_secret and the session_hash parameter which hashes together all handshake messages that have been sent or received (including ClientHello.random and ServerHello.random). This update prevents the man-in-the-middle attacks that may work, as explained in the Abstract section of RFC 7627, against the TLS protocols that rely on the original definition of the master secret parameter.

D.R Moved to W.2

Annex E – Approved authentication mechanisms

E.A Applicability of Requirements from SP 800-63B

Applicable Levels:	All
Original Publishing Date:	August 27, 2021
Effective Date:	August 27, 2021
Last Modified Date:	August 27, 2021
Relevant Assertions:	AS04.48
Relevant Test Requirements:	TE04.48.01
Relevant Vendor Requirements:	VE04.48.01

Background

ISO/IEC 19790:2012, Section 7.4.4:

The module **shall [04.48]** implement an approved authentication mechanism as referenced in Annex E.

SP 800-140E, CMVP Approved Authentication Mechanism – ***CMVP Validation Requirements for ISO/IEC 19790:2012 Annex E and ISO/IEC 24759 Section 6.17***.

Section 2, 'Normative References' identifies normative references as **ISO/IEC 19790:2012** and **ISO/IEC 24759:2017** exclusively.

Section 5.1, states the following "While this document serves a different purpose, much of the authentication is purposely meant to align with **SP 800-63B**, which is an informative reference for module authentication."

Section 6.2, 'Approved authentication mechanisms' includes the following statements:

"Vendors should use **SP 800-63B** as a framework for authentication requirements and should provide justification whenever **SP 800-63B** requirements cannot be met, or additional information is necessary. Testers should review and affirm the vendor documentation.

Normative **SP 800-63B** sections include

- Section 5, Detailed requirements specific to each type of authenticator;
- Section 6, Lifecycle management;
- Section 7, Session Management;

Informative information to be assessed for each authenticator includes:

- Section 8, Threats and Security Considerations; and
- Section 10, Usability Considerations."

Question/Problem

Is **SP 800-63B** a 'normative' or 'informative' reference to **SP 800-140E**?

When is it acceptable to justify non-compliance to requirements from **SP 800-63B**?

Resolution

SP 800-63B is informative and the list of documents covered in Section 2 of **SP 800-140E** are the exclusive list of normative references to **SP 800-140E**.

SP 800-140E, Section 6.2 states **SP 800-63B** sections 5, 6 and 7 are normative. Section 5.1 states **SP 800-63B** is an informative reference; therefore, these sections are not mandated for compliance to FIPS 140-3.

Sections 8 and 10 of **SP 800-63B** are recommendations not written in a format to easily determine what should be required.

Examples of valid justification where **SP 800-63B** requirements cannot be met could include:

1. target requirements can be shown to be beyond the scope of requirements on authentication set out by ISO/IEC 19790:2012 in section 7.4.4, 'Authentication' and by association, paragraph 1 and Table 1 from Section 6.2 of SP 800-140E.
2. requirements are not applicable based on the type of authentication mechanism deployed by the target cryptographic modules.
3. requirements are not applicable because the authentication mechanism deployed by the target cryptographic module is not for a human user.
4. the target cryptographic module is not to be deployed to an environment where users are bound to a mandated 'Authentication Assurance Level (AAL)' as covered in SP 800-63B.
5. threats underpinning a target requirement from SP 800-63B can be shown to be mitigated through other technical means by the cryptographic module.
6. threats underpinning a target requirement can be shown to not apply in the context of how a given cryptographic module is deployed and used.

The vendor and CST lab **shall** document in **AS04.48** how the module is aligned with sections of **SP 800-63B** or provide justification where **SP 800-63B** requirements cannot or should not be met.

Additional Comments

Strict compliance to **SP 800-63B** is not mandated for FIPS 140-3; however, compliance with applicable requirements is strongly encouraged.

Annex F – Approved non-invasive attack mitigation test metrics

Withdrawn Guidance

W.1 (was D.E) Assurance of the Validity of a Public Key for SSP establishment

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	September 21, 2020
Date Withdrawn	**March 17, 2023**
Relevant Assertions:	AS09.10
Relevant Test Requirements:	TE09.10.01-02
Relevant Vendor Requirements:	VE09.10.01

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 800-56Arev3** 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.

W.2 (was D.R) Hash Functions Acceptable for Use in the SP 800-90A DRBGs

Applicable Levels:	All
Original Publishing Date:	May 16, 2022
Effective Date:	May 16, 2022
Last Modified Date:	
Date Withdrawn	**August 30, 2024**
Transition End Dates	May 16, 2023 – See Below
Relevant Assertions:	AS02.20, AS09.06, AS09.09
Relevant Test Requirements:	TE(s) associated with AS(s) above
Relevant Vendor Requirements:	VE(s) associated with AS(s) above

Background

SP 800-90Arev1 introduces two kinds of the DRBG mechanisms: those based on hash functions and those based on block ciphers. The hash-function-based mechanisms include Hash_DRBG and HMAC_DRBG.

Table 2 in Section 10.1 of **SP 800-90Arev1** shows that the following hash functions can be used in the hash-based mechanisms: (1) SHA-1, (2) SHA-224 and SHA-512/224, (3) SHA-256 and SHA-512/256, (4) SHA-384, and (5) SHA-512. While there is no security exposure that may be caused by using any of the above hash functions in the DRBGs, it has been noted in **SP 800-90Arev1** that there is no efficiency improvement when using SHA-224 rather than SHA-256 and that there is no efficiency improvement when using SHA-384, SHA-512/224 or SHA-512/256 rather than SHA-512.

Question/Problem

Shall all hash functions listed above be allowed in the Hash_DRBG and HMAC_DRBG algorithms?

Resolution

The already-validated modules and those with the Test Reports submitted to the CMVP no later than **one year after this IG is published** can remain unchanged.

The new validations, or any revalidations that extend the module's sunset date, submitted more than one year after the publication date of this IG **shall** only use SHA-1, SHA-256 or SHA-512 in Hash_DRBG and HMAC_DRBG. The revalidations not extending the module's sunset date are not subject to this requirement.

Additional Comments

1. At the time this IG is published, there are no implementations of Hash_DRBG or HMAC_DRBG which use any of the SHA3 family hash functions. If such implementations appear in the future, they **shall** be using SHA3-256 or SHA3-512, and not SHA3-224 or SHA3-384.

W.3 (was C.G) SP 800-67rev2 Limit on the Number of Encryptions with the Same Triple-DES Key

Applicable Levels:	All
Original Publishing Date:	September 21, 2020
Effective Date:	September 21, 2020
Last Modified Date:	September 21, 2020
Date Withdrawn	**April 18, 2025**
Relevant Assertions:	AS02.20
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. The module can meet this AS for Triple-DES usage by 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.

Change Summary

New Guidance

- 09/10/24: C.O Requirements for SP 800-208 HSS Vendor Affirmation
- 07/26/24: 1.A Binding and Embedding Cryptographic Modules
- 07/26/24: C.M Legacy Algorithms
- 07/26/24: C.N Requirements for SP 800-208 schemes
- 07/25/23: 10.3.F Complete Image Replacement Versus Software/Firmware Loading
- 07/25/23: C.K Transition from FIPS 186-4 to FIPS 186-5 and SP 800-186
- 07/25/23: C.L SP 800-107 Requirements
- 03/17/23: 10.2.A Pre-operational Integrity Technique Self-test
- 03/17/23: 2.3.D Excluded Components
- 05/16/22: D.Q Transition of the TLS 1.2 KDF to Support the Extended Master Secret
- 05/16/22: D.R Hash Functions Acceptable for Use in the SP 800-90A DRBGs
- 11/05/21: D.P SP 800-56Crev2 One-Step Key Derivation Function Without a Counter
- 08/27/21: 10.3.D Error Logging
- 08/27/21: 10.3.E Periodic Self-Testing
- 08/27/21: E.A Applicability of Requirements from SP 800-63B
- 05/04/21: 2.4.C Approved Security Service Indicator
- 05/04/21: 9.7.B Indicator of Zeroisation
- 05/04/21: 10.3.C Conditional Manual Entry Self-Test Requirements
- 05/04/21: 11.A CVE Management
- 05/04/21: 12.A Mitigation of Other Attacks
- 05/04/21: D.O Combining Entropy from Multiple Sources
- 09/21/20 - Initial release

Modified Guidance

- 04/18/25: Cleaned up IG to reflect the Triple-DES transition in Jan 2024 (i.e., admin updates to 2.4.A, 9.6.A, 10.3.A and the withdrawal of C.G in W.3). Where applicable, updated FIPS 140-3 Management Manual and WebCryptik User's Guide references.
- 04/18/25: 9.5.A SSP Establishment and SSP Entry and Output – Admin update to footnote 3 to point to IG D.G and D.F instead of SP 800-140D.
- 04/18/25: 10.3.A Cryptographic Algorithm Self-Test Requirements – Admin update to FIPS 203 (ML-KEM), FIPS 204 (ML-DSA) and FIPS 205 (SLH-DSA) self-tests to permit self-testing the internal algorithms in lieu of testing the external algorithms and updated to ensure these algorithm parameter references are consistent/correct (e.g., added parameters

for the internal algorithms and added the missing context string, *ctx*, as applicable). Modified Note23 to change to a 'should' instead of a 'shall' given some challenges testing all the rejection paths.

- 04/18/25: 10.3.F Complete Image Replacement Versus Software/Firmware Loading – Admin update to Additional Comment 7 to include trusted anchor.
- 04/18/25: C.K Transition from FIPS 186-4 to FIPS 186-5 and SP 800-186 – Admin update to improve clarity in Additional Comment #2 regarding FIPS 186-4 CAVP tests.
- 04/18/25: D.B Strength of SSP Establishment Methods – Admin update to correct SP 800-57 table references.
- 12/20/24: C.M Legacy Algorithms – Revised "Symmetric Algorithms Used for Decryption / Unwrapping" to break out rows for clarity and include unauthenticated AES. Minor clean up in other areas of the IG.
- 10/23/24: 2.3.C Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI) – Reworked Additional Comment #3, as it was confusing/misleading. Removed "software" from Additional Comment #5 regarding modules that support a PAI since it may apply to other module types. Added a new Additional Comment #7 to clarify that PAA/PAIs operate within a processor.
- 10/21/24: 2.3.C Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI) – Updated Additional Comment #4 on what is required to request new "known" PAA or PAIs to the list.
- 10/21/24: 10.3.A Cryptographic Algorithm Self-Test Requirements – Merged Resolution 3 (FIPS 202) and 4 (SP 800-185) for consistency.
- 10/21/24: C.C The Use and the Testing Requirements for the Family of Functions defined in FIPS 202 – Added references to FIPS 186-5 and SP 800-208.
- 10/21/24: C.H Key/IV Pair Uniqueness Requirements from SP 800-38D – Broke out the restoration conditions into Scenario 3a to be independently referenceable by the other scenarios.
- 10/21/24: D.C References to the Support of Industry Protocols – Added a new reference to the OTAR specification, published August 2023, that incorporates AES CMAC.
- 09/10/24: C.N Requirements for SP 800-208 Schemes – Added reference to C.O for vendor affirming HSS.
- 08/30/24: W.2 Hash Functions Acceptable for Use in the SP 800-90A DRBGs – IG Withdrawn.
- 08/30/24: Editorial fix to correct numbering issues for several IGs.
- 08/14/24: Editorial fix to correct Additional Comments not being numbered for several IGs.
- 08/13/24: 10.3.A Cryptographic Algorithm Self-Test Requirements – Added self-test requirements for the Post Quantum Algorithms (PQC) specified in FIPS 203, FIPS 204, and FIPS 205. Added general statement on the comparison test and removed references to the comparison and fault detection alternatives since they are unpractical in most cases. Updated Additional Comment 1 to clarify when the PCT in the underlying standard is implemented. Clarified the SP 800-90B health-tests should be categorized as fault detection tests per AS10.34. Other minor editorial changes.
- 08/13/24: C.I XTS-AES (SP 800-38E) Requirements on the Key – Name change to better reflect the topic covered by this IG.
- 07/26/24: 2.3.C Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI) – Added Additional Comment #6 regarding ESV and PAI/PAAs.
- 07/26/24: 2.4.B Tracking the Component Validation List – Improved the details for CVL tests and mapped them to the latest CAVP tests. Introduced usage restrictions for CVLs. Removed

outdated Additional Comment 5 and moved Additional Comment 4 into the applicable Resolution of this IG.

- 07/26/24: 9.3.A Entropy Caveats – Updated to require ESV when the source is within the OE, passive or not. Added Resolution 3 (therefore pushing the Hybrid case to Resolution 4) disallowing scenarios where the module does not have direct access to the entropy source's GetEntropy() interface. Added related Additional Comments #9, #10, #11 and #12. Updated Additional Comment #4 to clarify the claimed security strength of any approved algorithm may not be greater than 256 bits, which impacts the applicability of certain entropy caveats.
- 07/26/24: 10.3.A Cryptographic Algorithm Self-Test Requirements – Moved Additional Comment #2 and #3 into the Resolution section. Labeled the ending requirements in the Resolution as "General CAST Requirements" and included numbered Notes.
- 07/26/24: C.F RSA Approved Parameter Sizes in FIPS 186-5 – Name change and removal of Additional Comments #1, #2, and #3 to better reflect the topics this IG covers. Added a strong recommendation (but not requirement) regarding Miller-Rabin test requirements in the Resolution and adjusted the related Additional Comments #1 and #2 (was #4 and #5). Revised to target compliance to FIPS 186-5. Added a rationale for the FIPS 186-5 restrictions on the primes' sizes. In the previous version of this IG, the RSA auxiliary primes size recommendations were different from what the digital signature standard then in place (FIPS 186-4) was saying and the IG had to provide the justification for the discrepancy.
- 07/26/24: C.K Transition from FIPS 186-4 to FIPS 186-5 and SP 800-186 – Slight update to Additional Comment #2 to clarify when FIPS 186-5 CAVP tests are required.
- 07/26/24: D.F Key Agreement Methods – Moved C.F Additional Comment #2 to D.F Additional Comment #12.
- 03/26/24: 2.3.B Sub-Chip Cryptographic Subsystems – small correction to the paragraph that references IG 9.5.A.
- 03/26/24: 4.1.A Authorised Roles – Updated Additional Comment #8 to address certain module designs that claim Security Level 2 for section 7.4.
- 03/26/24: 9.5.A SSP Establishment and SSP Entry and Output – Added footnote 6 to clarify sub-chip SSP establishment requirements.
- 03/26/24: C.A Use of non-Approved Elliptic Curves – Removed erroneous reference to EdDSA from Resolution 5.
- 03/26/24: 10.3.A Cryptographic Algorithm Self-Test Requirements – Added new footnote #1 to Additional Comment #1 to specify that an RSA PCT which satisfies TE10.35.02 also satisfies TE10.35.01.
- 01/29/24: 10.3.A Cryptographic Algorithm Self-Test Requirements – Added **Note20** to clarify the TLS KDF self-test requirements.
- 01/29/24: C.K Transition from FIPS 186-4 to FIPS 186-5 and SP 800-186 – Resolution 4: K and B curves will be included in the FIPS 186-5 testing. Resolution 6: Removed specific reference to P curves since this ECDSA verification using K and B curves is also approved. Additional Comment 2: Clarified that mathematically equivalent FIPS 186-4 tests can claim FIPS 186-5 compliance.
- 01/29/24: D.C References to the Support of Industry Protocols – Added Additional Comment #2 specifying that this IG includes the TLS 1.3 KDF CVL.
- 11/22/23: 2.3.C Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI) – Added a few Known PAAs.

- 11/22/23: 2.4.C Approved Security Service Indicator - Clarified the API example in the Resolution and added a related Additional Comment 5.
- 11/22/23: 4.1.A Authorised Roles - Added "[for CSPs only]" in Background. Clarified in a. the exception applies when hashing data, not SSPs. Added a paragraph after the exceptions connecting authorization to authentication.
- 11/22/23: 9.5.A SSP Establishment and SSP Entry and Output - Slight modification to the SK legend under Table 2.
- 11/22/23: C.C The Use and the Testing Requirements for the Family of Functions defined in FIPS 202 - Removed the outdated Additional Comments.
- 11/22/23: C.H Key/IV Pair Uniqueness Requirements from SP 800-38D - Changed "technique" to "scenario" in the beginning of the Resolution for consistency. Added leniency to the abort logic requirement in Scenario 3.
- 08/01/23: D.B Strength of SSP Establishment Methods – Removed outdated text regarding how to document the SSP establishments on the certificate.
- 07/25/23: 2.4.C Approved Security Service Indicator – Added Additional Comment #4 to clarify the applicability of example scenarios 1) and 3). Updated the first bullet after "IG clarifies **AS02.24** by interpreting the following:" to align closer to TE02.24.01 and TE.02.24.02.
- 07/25/23: 10.3.A Cryptographic Algorithm Self-Test Requirements – Added self-test requirements for FIPS 186-5 algorithms. Clarified self-test requirements for underlying approved algorithms used within a higher-level algorithm with examples. Added Additional Comment #3 on general self-test requirements. Some formatting and editorial changes.
- 07/25/23: C.A Use of non-Approved Elliptic Curves - Removed Additional Comment #1 since the transition is now published. Revised Additional Comment #2 (now #1) to specify EdDSA status. Incorporated final draft guidance from IG C.K into Category 1a and 1b.
- 07/25/23: D.G Key Transport Methods – Updated Additional Comment #4 to be consistent with WebCryptik and CAVP representation (e.g., KTS-IFC).
- 07/25/23: D.F Key Agreement Methods – Updated Additional Comment #5 to clarify requirements for assurances. Updated KAS references be consistent with WebCryptik and CAVP representation (i.e., KAS-ECC or KAS-FFC).
- 07/25/23: Added reference to FIPS 186-5 in addition to or instead of FIPS 186-4. This resulted in minor admin changes (published date remained unchanged) to IGs:
  - 2.4.A Definition and Use of a non-Approved Security Function
  - 4.1.A Authorised Roles
  - D.B Strength of SSP Establishment Methods
  - D.D Elliptic Curves and the FFC Safe-Prime Groups in Support of Industry Protocols
- 07/25/23: Updated to reference ESV. This resulted in minor admin changes (published date remained unchanged) to IGs:
  - D.J Entropy Estimation and Compliance with SP 800-90B
  - D.O Combining Entropy from Multiple Sources
- 03/17/23: Entire IG – Updated FIPS 140-3 Management Manual references (several replaced by WebCryptik User's Guide) and revalidation scenario references.
- 03/17/23: W.1 Assurance of the Validity of a Public Key for SSP establishment – IG Withdrawn.
- 03/17/23: 2.3.B Sub-Chip Cryptographic Subsystems – Updated Note 2 references to TE02.13.03. Removed porting guidance (moved to FIPS 140-3 Management Manual Section 7.1). Added Additional Comment #3 on validation status.

- 03/17/23: 2.3.C Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI) – Fixed PAA/PAI bulleted examples.
- 03/17/23: 4.1.A Authorised Roles – Added SP 800-90B under Resolution b. Added Additional Comments #6, #7, and #8.
- 03/17/23: 5.A Non-Reconfigurable Memory Integrity Test – Added reference to TE02.03.02 in the Resolution.
- 03/17/23: 9.3.A Entropy Caveats – Updated caveats to include "(e.g., keys)" in the SSP references.
- 03/17/23: 10.3.A Cryptographic Algorithm Self-Test Requirements – Added SP 800-208 self-test requirements (Note: SP 800-208 algorithms can only be used in the approved mode if certified by the CAVP, once testing becomes available). Clarified SP 800-90B self-tests are considered CASTs. Clarified self-test requirements for algorithms whose output vary for a given set of inputs. Added SSH KDF and IKE KDF self-tests when used within an approved KAS. Aligned vendor affirmed self-test guidance with FIPS 140-3 Management Manual. Updated Additional Comment #1 on the key-pair PCT requirements.
- 03/17/23: C.H Key/IV Pair Uniqueness Requirements from SP 800-38D – Added references to DTLS 1.2 in Scenario 1.
- 03/17/23: D.F Key Agreement Methods – Added Additional Comment #11 to clarify CVL KDF CAST requirements.
- 03/17/23: D.H Requirements for Vendor Affirmation to SP 800-133 – Added Additional Comment #5 on CAST requirements.
- 03/17/23: D.K Interpretation of SP 800-90B Requirements – Added headers to group the Resolutions and added Resolution 19 on full entropy. Added requirements when a DRBG is considered a conditioning component (updates to Resolution 5 and Resolution 7 Note 1).
- 03/17/23: D.Q Transition of the TLS 1.2 KDF to Support the Extended Master Secret – Updated Additional Comment #1 on TLS 1.0 and TLS 1.1 KDFs and their transition when using the extended master secret.
- 10/07/22: 2.3.C Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI) – Clarified the testing requirements when a module incorporates PAA or PAI functionality. Updated known PAA/PAIs.
- 10/07/22: 9.3.A Entropy Caveats – Added Additional Comment #7 on claiming multiple scenarios from this IG and added Additional Comment #8 on which scenarios require an entropy assessment report.
- 10/07/22: C.F - Approved Modulus Sizes for RSA Digital Signature for FIPS 186-4 – Clarified algorithm status and requirements for RSA Signature Verification for both FIPS 186-2 and FIPS 186-4.
- 10/07/22: Mapping FIPS 140-2 IGs to FIPS 140-3 – Added a mapping to the FIPS 140-3 Management Manual section.
- 05/16/22: 3.4.A Trusted Channel – Removed Additional Comment #2 as this is appropriate for FIPS 140-2, but does not align with requirements of **ISO/IEC 19790:2012** Section 7.9.5 and IG 9.5.A.
- 05/16/22: 9.5.A SSP Establishment and SSP Entry and Output – Added parenthesis to highlight the fact that there are differences in requirements between CSPs that are keys versus non-keys.
- 03/14/22: 2.4.A Definition and Use of a non-Approved Security Function – Added "with no security claimed" to the examples subtitle for clarity. Small editorial change in the Resolution to reference the correct algorithm table in **SP 800-140B**. Added a footnote to MD5.
- 03/14/22: 2.4.B Tracking the Component Validation List – Added vendor affirmation of a SRTP KDF implementation.

- 11/05/21: Added a space to all ENT entries to ENT (P) or ENT (NP).
- 11/05/21: 2.4.B Tracking the Component Validation List – Added references to SP 800-56Arev3 for the ECC-CDH primitive CVL in Resolution #1.
- 11/05/21: 2.4.A Definition and Use of a non-Approved Security Function – Synchronized minor text in the Resolution to be consistent with IG 1.23 (FIPS 140-2). Clarified XOR example with a note. Added Additional Comment #2 to further clarify when a vendor can apply this IG.
- 11/05/21: 10.3.A Cryptographic Algorithm Self-Test Requirements – Spelled out the ENT self-test requirements to avoid ambiguity.
- 11/05/21: C.F Approved Modulus Sizes for RSA Digital Signature for FIPS 186-4 – Added Table 1 with a more relaxed upper bound limit and introduced supporting text including adding two new Additional Comments. Clarified the minimum number of the Miller-Rabin tests. Cleaned up old text in the Additional Comments.
- 11/05/21: D.C References to the Support of Industry Protocols – Included guidance on the use of AES-CBC-MAC within OTAR.
- 11/05/21: D.J Entropy Estimation and Compliance with SP 800-90B – Added Additional Comment #10 to clarify when other parties can write a labs' entropy source description and its heuristic entropy analysis.
- 11/05/21: D.L Critical Security Parameters for the SP 800-90A DRBGs – Added Additional Comment on the CTR_DRBG without a derivation function.
- 08/27/21: 5.A Non-Reconfigurable Memory Integrity Test – Incorporated end of life procedures.
- 05/04/21: 3.4.A Trusted Channel – Clarified in the last bullet in Resolution 2 that the operator must stay in control over the physical path and prevent any unauthorised tampering.
- 05/04/21: 4.1.A Authorised Roles - Clarified the requirements of the text "or other services that do not affect the security of the module".
- 05/04/21: 5.A Non-Reconfigurable Memory Integrity Test – 20 years was changed to 10 years.
- 05/04/21: 10.3.A Cryptographic Algorithm Self-Test Requirements – Updated to remain consistent with FIPS 140-2 IG 9.4. Also, clarified self-test rules around the PBKDF Iteration Count parameter.
- 05/04/21: C.H Key/IV Pair Uniqueness Requirements from SP 800-38D - Removed Scenario 2's second and fourth bullets and added the reasoning as Additional Comment #4.
- 05/04/21: D.F Key Agreement Methods - Removed Additional Comment 10 since SP 800-56Arev3 testing is available and therefore vendor affirming to this standard is not permitted.
- 05/04/21: D.G Key Transport Methods - Added "if applicable" for key confirmation under the first approved method.
- 05/04/21: D.J Entropy Estimation and Compliance with SP 800-90B - Updated to align ENT references with that of IG D.O.

Mapping FIPS 140-2 IGs to FIPS 140-3

FIPS 140-2 IG	FIPS 140-3 Management Manual
G.2 Completion of a test report: Information that must be provided to NIST and CCCS	WebCryptik User's Guide (see SP 800-140B supplemental documents)
G.13 Instructions for Validation Information Formatting	CMVP webpage: Caveats and SP 800-140B supplemental documents
G.1 Request for Guidance from the CMVP and CAVP	2.5 Request for Guidance from CMVP
G.7 Relationships Among Vendors, Laboratories, and NIST/CCCS	2.5 Request for Guidance from CMVP
G.12 Post-Validation Inquiries	2.5.3 Post Validation Inquiries
G.4 Design and testing of cryptographic modules	2.6.2 Cryptographic and Security Testing Laboratory
G.9 FSM, Security Policy, User Guidance and Crypto Officer Guidance Documentation	2.6.2 Cryptographic and Security Testing Laboratory
G.16 Requesting an Invoice Before Submitting a Report	4.6.4 Invoice for a Report Submission
G.8 Revalidation Requirements	7.1 Submission Scenarios
A.3 Vendor Affirmation of Cryptographic Security Methods	7.2 CMVP requirements pertaining to testing and approved algorithms
G.11 Testing using Emulators and Simulators	7.3 Testing using Emulators and Simulators
G.17 Remote Testing for Software Modules	7.4 Remote Testing of Software Modules
G.3 Partial Validations and Not Applicable Areas of FIPS 140-2	7.5 Partial validations and non-applicable areas
1.18 PIV Reference	7.6 CMVP requirements for PIV validations
1.22 Module Count Definition	7.7 Module count definition
G.5 Maintaining validation compliance of software or firmware cryptographic modules	7.9 Vendor or User Affirmation of Modules
G.19 Operational Equivalency Testing for HW Modules	7.10 Operational Equivalency Testing for HW Modules

FIPS 140-2 IG	FIPS 140-3 IG
1.4 - Binding of Cryptographic Algorithm Validation Certificates	2.3.A - Binding of Cryptographic Algorithm Validation Certificates
1.20 - Sub-Chip Cryptographic Subsystems	2.3.B - Sub-Chip Cryptographic Subsystems
1.21 - Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI)	2.3.C - Processor Algorithm Accelerators (PAA) and Processor Algorithm Implementation (PAI)
1.23 - Definition and Use of a non-Approved Security Function	2.4.A - Definition and Use of a non-Approved Security Function
G.20 - Tracking the Component Validation List	2.4.B - Tracking the Component Validation List
2.1 - Trusted Path	3.4.A - Trusted Channel
3.1 - Authorized Roles	4.1.A - Authorised Roles
3.4 - Multi-Operator Authentication	4.4.A - Multi-Operator Authentication
9.13 - Non-Reconfigurable Memory Integrity Test	5.A - Non-Reconfigurable Memory Integrity Test
5.2 - Testing Tamper Evident Seals	7.3.A - Testing Tamper Evident Seals
5.4 - Level 3: Hard Coating Test Methods	7.3.B - Hard Coating Test Methods (Level 3 and 4)
7.14 - Entropy Caveats	9.3.A - Entropy Caveats
7.7 - Key Establishment and Key Entry and Output	9.5.A - SSP Establishment and SSP Entry and Output
7.16 - Acceptable Algorithms for Protecting Stored Keys and CSPs	9.6.A - Acceptable Algorithms for Protecting Stored SSPs
7.17 - Zeroization of One Time Programmable (OTP) Memory	9.7.A - Zeroisation of One Time Programmable (OTP) Memory
9.4 - Known Answer Tests for Cryptographic Algorithms	10.3.A - Cryptographic Algorithm Self-Test Requirements
9.2 - Known Answer Test for Embedded Cryptographic Algorithms	10.3.B - Self-test for Embedded Cryptographic Algorithms
11.1 Mitigation of Other Attacks	12.A - Mitigation of Other Attacks
A.2 - Use of non-NIST-Recommended Asymmetric Key Sizes and Elliptic Curves	C.A - Use of non-Approved Elliptic Curves

A.1 - Validation Testing of SHS Algorithms and Higher Cryptographic Algorithm Using SHS Algorithms	C.B - Validation Testing of Hash Algorithms and Higher Cryptographic Algorithm Using Hash Algorithms
A.11 - The Use and the Testing Requirements for the Family of Functions defined in FIPS 202	C.C - The Use and the Testing Requirements for the Family of Functions defined in FIPS 202
A.8 - Use of a Truncated HMAC	C.D - Use of a Truncated HMAC
7.12 - Key Generation for RSA Signature Algorithm	C.E - Key Generation for RSA Signature Algorithm
A.14 - Approved Modulus Sizes for RSA Digital Signature and Other Approved Public Key Algorithms	C.F - Approved Modulus Sizes for RSA Digital Signature for FIPS 186-4
A.13 - SP 800-67rev1 Transition	C.G - SP 800-67rev2 Limit on the Number of Encryptions with the Same Triple-DES Key
A.5 - Key/IV Pair Uniqueness Requirements from SP 800-38D	C.H - Key/IV Pair Uniqueness Requirements from SP 800-38D
A.9 - XTS-AES Key Generation Requirements	C.I - XTS-AES (SP 800-38E) Requirements on the Key
A.10 - Requirements for Vendor Affirmation of SP 800-38G	C.J - Requirements for Testing to SP 800-38G
D.2 - Acceptable Key Establishment Protocols	D.A - Acceptable SSP Establishment Protocols
7.5 - Strength of Key Establishment Methods	D.B - Strength of SSP Establishment Methods
D.11 - References to the Support of Industry Protocols	D.C - References to the Support of Industry Protocols
D.13 - Elliptic Curves and the MODP Groups in Support of Industry Protocols	D.D - Elliptic Curves and the FFC Safe-Prime Groups in Support of Industry Protocols
D.3 - Assurance of the Validity of a Public Key for Key Establishment	D.E - Assurance of the Validity of a Public Key for SSP establishment
D.8 - Key Agreement Methods	D.F - Key Agreement Methods
D.9 - Key Transport Methods	D.G - Key Transport Methods
D.12 - Requirements for Vendor Affirmation to SP 800-133	D.H - Requirements for Vendor Affirmation to SP 800-133
7.8 - The Use of Post-Processing in Key Generation Methods	D.I - The Use of Post-Processing in Key Generation Methods

7.18 - Entropy Estimation and Compliance with SP 800-90B	D.J - Entropy Estimation and Compliance with SP 800-90B
7.19 - Interpretation of SP 800-90B Requirements	D.K - Interpretation of SP 800-90B Requirements
14.5 - Critical Security Parameters for the SP 800-90 DRBGs	D.L - Critical Security Parameters for the SP 800-90A DRBGs
7.10 - Using the SP 800-108 KDFs in FIPS Mode	D.M - Using the SP 800-108 KDFs in an Approved Mode
D.6 - Requirements for Vendor Affirmation of SP 800-132	D.N - SP 800-132 Password-Based Key Derivation for Storage Applications
7.20 - Combining Multiple Entropy Sources	D.O - Combining Multiple Entropy Sources
D.14 - SP 800-56C Rev2 One-Step Key Derivation Function Without a Counter	D.P - SP 800-56Crev2 One-Step Key Derivation Function Without a Counter

End of Document