centro criptológico nacional

Guía de Seguridad de las TIC
CCN-STIC 2100

Cryptographic Mechanisms Evaluation Methodology
Metodología de Evaluación de Mecanismos Criptográficos (MEMeC)

Noviembre de 2024

MINISTERIO DE DEFENSA

Catálogo de Publicaciones de la Administración General del Estado
https://cpage.mpr.gob.es

Edita:

MINISTERIO DE DEFENSA
SUBSECRETARÍA DE DEFENSA
SECRETARÍA GENERAL TÉCNICA

Pº de la Castellana 109, 28046 Madrid

Fecha de Edición: noviembre de 2024
NIPO: pendiente de asignación.

CENTRO CRIPTOLOGICO NACIONAL
cn=CENTRO CRIPTOLOGICO NACIONAL, 2.5.4.97=VATES-S2800155J, ou=CENTRO CRIPTOLOGICO NACIONAL, o=CENTRO CRIPTOLOGICO NACIONAL, c=ES
2024.11.27 17:28:14 +01'00'

TABLE OF CONTENTS

INDEX OF TABLES

INDEX OF FIGURES

1 INTRODUCTION

1. The use of cryptography is becoming more and more present in technological products due to the necessity of protecting the information to ensure its confidentiality, integrity, and authenticity. Thus, it is necessary to verify the cryptographic mechanisms implemented on these technological products to guarantee their correct operativity and their correct usage and implementation by the vendors.

1.1 OBJECTIVE

2. The objective of this document is to define a common methodology to evaluate the cryptographic implementations included in technological products that are going to be evaluated under the Common Criteria (CC), LINCE, or STIC certifications according to the security requirements related to the certification levels defined in the following sections.
3. This methodology provides the steps to verify that the cryptographic mechanisms implemented in the Target Of Evaluation (TOE) and its parameterization comply with the requirements defined by the Centro Criptológico Nacional (CCN). Moreover, this methodology defines the steps to check the correct operativity of each cryptographic mechanism employing test vectors as input to carry out the conformity testing for each of them.
4. On top of providing the steps to check the correct parameterization and operativity of each cryptographic mechanism, this methodology also defines how to check that each of them is implemented and executed avoiding common implementation pitfalls that may cause misbehavior on the cryptographic mechanism, and that could incur in revealing sensitive information such as user data and Sensitive Security Parameters (SSPs).
5. Finally, the methodology includes specific tasks to verify that the TOE checks the correct operativity of the implemented cryptographic mechanisms using self-tests, that the SSP management performed by the TOE complies with the CCN requirements and is based on approved cryptographic mechanisms, and that the TOE implements additional protections for the mitigation of other attacks.

1.2 STRUCTURE OF THE DOCUMENT

6. The definition of the cryptographic requirements to be met by the TOE and the evaluation tasks to be performed by the tester has been made following this structure:
   - The cryptographic requirements start with the words “The **TOE shall”** followed by the description of the requirement to be met by the TOE.
   - The evaluation tasks are defined in a table composed of two main parts.

- The first part specifies an evaluation task ID and a brief description of the tasks to be performed by the tester, which begins with the words "The **tester shall"** followed by a brief description and the required inputs by the tester to carry out the evaluation. The task ID is denoted using bold typography with the following format **CCN-NAME**, where "NAME" is the name assigned to the task.
- The second part defines the certification levels for which the evaluation task shall be executed. The format of an evaluation task is as detailed below:

CRYPTOGRAPHIC EVALUATION TASK	CCN-NAME	CERTIFICATION LEVEL
The **tester shall…** [Cryptographic Evaluation Task Definition]		**CL1, CL2, and/or CL3**

7. The cryptographic tests associated with each evaluation task are defined in a table divided into two columns.
   - The first column contains the identifier (ID) of the test and a brief description of the activities to be performed by the tester.
   - The second column specifies the inputs required to perform the test. The ID is written in bold typography following the format **CCN-NAME/TestName**, where "CCN-NAME" is the ID of the evaluation task associated with the test, and "TestName" is the specific name of the cryptographic test to be performed:

CCN-NAME/TestName	Inputs
[Cryptographic Evaluation Test Definition]	

1.3 CERTIFICATION LEVELS

8. This methodology defines three increasing certification levels:
   - ✓ **Certification Level 1 (CL1):** This is the lowest certification level. At this level, the main objective is to verify the compliance of the cryptographic primitives and constructions implemented in the TOE according to the [CCN-STIC-221] guidance. Therefore, it only requires performing the **CCN-AGREED** and **CCN-PROTOCOL** evaluation tasks and verifying the correct operativity of the cryptographic mechanisms by executing the **CCN-CONFORMITY** evaluation task.
   - ✓ **Certification Level 2 (CL2):** This is the intermediate certification level. At this level, all the evaluation tasks defined in this methodology shall be performed.

Regarding the Management of SSPs related to the **CCN-SSP** evaluation task, this certification level only requires evidence of zeroization of the SSPs managed by the TOE.

- ✓ **Certification Level 3 (CL3):** This is the most advanced certification level. At this level, all the evaluation tasks defined in this methodology shall be performed. The difference with the **CL2** level is that, in this case, evidence of the entire lifecycle of the SSPs (from its creation to its destruction) is required to perform the **CCN-SSP** evaluation task.

1.4 INPUTS

9. To simplify the revision process of the documentation and to reduce the evaluation time of the evaluation tasks defined in this methodology, the vendor shall provide the following inputs as part of the documentation required to perform the cryptographic mechanisms evaluation:

I1: Cryptographic Mechanisms Evaluation Vendor Questionnaire and, optionally, Vendor Documentation

10. The documentation package provided by the vendor is composed of:

    - **Vendor Questionnaire (VQ)**: a document with answers to a battery of questions provided by the laboratory in order to extract the main information related to the cryptographic mechanisms implemented by the TOE. These questions aim to demonstrate that the implementation of the cryptographic mechanisms (primitives and constructions) complies with the cryptographic requirements detailed throughout this methodology. For those cases for which the space provided in the VQ is not enough, the vendor can provide the required information by adding annexes to the VQ. In the case of **CL1** evaluations, a lite version is available where only the evaluation tasks that apply to this level are contemplated.
    - **Vendor Documentation**: the vendor documentation may encompass any document supplied by the vendor to provide additional information to that contained in the Vendor Questionnaire. This information may be used by the vendor to support the information specified in the VQ, but in no case it will be a substitute since it will not be considered part of the evaluation.

I2: Test and operation interfaces to verify the TOE functionality

11. On the one hand, the vendor shall specify operational interfaces that allow the tester to verify the correct operation of the TOE according to the cryptographic implementation described in **I1**.

12. On the other hand, the vendor shall also specify how the cryptographic mechanisms can be addressed, allowing the submission of input values, such as the key value in the case of keyed mechanisms. For randomized mechanisms, this testing interface may also possibly allow the submission of values to be used instead of the output of a random generator. The exact formatting conventions

for the external interface of the mechanism implementation of the system under evaluation, e.g., whether the input and output parameters are represented as binary strings or byte strings, shall be provided. **I2** also specifies how to collect the corresponding output values. For all mechanisms of the system under evaluation, this interface shall allow the tester to directly access the primitives and not only the modes.

13. These interfaces can either be operational interfaces or testing interfaces that might not be implemented in the final design to address the cryptographic mechanisms. In fact, such interfaces might be deliberately unavailable for security considerations. It is not mandatory to provide such interfaces in the final system under evaluation. A modified system that permits access to such interfaces is acceptable input. Sub-systems containing such interfaces, before the system integration, also constitute an acceptable input. In either case, a detailed description of the differences with the final system is necessary to justify that the tests will be valid in the final system.

I3: Conformity testing tools

14. The tester is responsible for verifying the correct operation of the cryptographic mechanisms implemented in the TOE by executing the conformity tests associated with each cryptographic mechanism (primitives and constructions), according to Section 4 Conformity Testing, to obtain the response files detailed in **I4**.

15. Taking into account that the evaluation time is limited, the vendor shall provide the tester with the necessary test environment to demonstrate the conformity of the cryptographic mechanisms. This test environment shall be composed of a set of software/firmware tools (e.g., test harnesses, scripts) that will allow the tester to generate and validate the results from each cryptographic mechanism using the conformity test vectors as input.

16. In addition, the vendor shall provide the tester with the necessary guidelines to properly configure and execute the test environment.

17. There may be some specific cases where the design and architecture of the TOE difficult the possibility to provide the tester with scripts to verify the conformity testing results. In such cases, on-site audits or telematic sessions will be allowed in order to demonstrate to the tester that the TOE successfully performs the conformity test.

I4: RSPs obtained from the conformity testing

18. The RSPs are the response files composed of the results obtained from the TOE when the conformity test vectors are used as input for each cryptographic mechanism. They shall be obtained using the test vectors generated by the laboratory. They will be the result of the conformity testing process. For more details, please consult Section 4 Conformity Testing.

I5: Evidence of the avoidance of implementation pitfalls

19. The vendor shall provide evidence of the avoidance of the implementation pitfalls associated with each cryptographic mechanism implemented in the TOE. The evidence shall consist of a piece of source code for **CL2** evaluations or pointers to **I6** for **CL3** evaluations.

I6: Implementation representation

20. For each cryptographic mechanism of the system under evaluation, the implementation representation is the most abstract representation of the mechanism. The representation is used to create the implementation. It may be software source code, firmware source code, hardware diagrams, and/or hardware design language code or layout data. Source code annotations are also required to help the tester establish the correspondence between the specification and the implementation.

21. If the cryptographic implementation is provided by an external library, **I6** shall include the source code that calls the library functions. This allows the tester to analyze the correctness of the library calls.

22. Additional information such as compilers, compilation options, and scripts are part of the implementation if they contribute to the generation of the mechanism implementation. Programming language specifications shall be provided. Compilation configuration files may also be part of the evaluation inputs if they contribute to the implementation of cryptographic mechanisms.

I7: Self-Test input vectors

23. The tester is responsible for verifying the validity of the self-test input vectors implemented for each cryptographic mechanism using the CCN Cryptographic Tool as the cryptographic reference implementation.

24. To perform this validation process, the tester shall generate a self-test template file in JSON format for each cryptographic mechanism implemented in the TOE, containing the parameters associated with the supported configuration. The vendor shall complete these templates with the test vectors used in each of the cryptographic mechanisms self-tests implemented.

25. Finally, the tester shall use the self-test files to perform the validation of the input vectors using the CCN Cryptographic Tool. For more details, please consult Section 2.3.1.3 “Cryptographic Mechanisms Self-Test”.

I8: Random Number Generation Vendor Questionnaire

26. In this document, the vendor shall include all the necessary evidence to evaluate the compliance of the TRNGs and DRNGs implemented by the TOE in accordance with the AIS-31 and AIS-20 functionality classes respectively, agreed in this methodology.

27. The document shall include all the evidence needed by the tester to evaluate all the requirements described in the [FCRNG] document for each functionality class.

28. This document follows the same philosophy as **I1** but for the RNG design implemented by the TOE.

1.5 EVALUATION PROCESS

29. This section describes the steps to be followed by the tester to perform a cryptographic mechanisms evaluation once the inputs detailed above have been received from the vendor.

30. The **tester shall** go through the sections of this document executing the cryptographic evaluation tasks associated with the certification level related to the TOE evaluation. Considering that each evaluation task is composed of one or several tests, the **tester shall** execute those categorized as mandatory and those categorized as Implementation-Dependent (Impl-Dep), only in case the TOE supports the associated functionality. At the beginning of each section, a summary table is provided specifying the cryptographic tests of each evaluation task, along with their categorization. The table format is detailed below, the first column indicates the associated evaluation task, the second column the test name, and the last column the test categorization:

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
CCN-NAME	CCN-NAME/TestName	Mandatory or Impl-Dep

31. To determine which Implementation-Dependent tests related to the cryptographic mechanisms shall be executed, the **tester shall** verify **I1** to identify the cryptographic primitives and constructions implemented by the TOE, and use the summary tables defined in “Annex A: Traceability of Evaluation Tasks”.

32. Following this scheme, the **tester shall** verify that the implemented cryptographic mechanisms meet the cryptographic requirements defined by CCN in Section 2 “CCN Cryptographic Requirements” and that they are considered agreed upon and use a valid parameterization according to Section 3 “CCN Agreed Cryptographic Mechanisms”.

33. In the case of CCN-approved cryptographic mechanisms that employ a valid parameterization, the **tester shall** verify their correct operation by executing the conformity tests, as detailed in Section 4 “Conformity Testing” using the CCN Cryptographic Tool as the cryptographic reference implementation. In addition, the **tester shall** verify that the cryptographic mechanisms have been implemented avoiding the common implementation pitfalls defined in Section 5 “Avoidance of Implementation Pitfalls”.

34. Following the same approach as for the cryptographic mechanisms, the **tester shall** use **I1** and the summary tables defined in **"**Annex A: Traceability of Evaluation Tasks**"** to identify the cryptographic evaluation tests that shall be executed regarding SSP management, implementation of self-tests, and mitigation of other attacks.

2 CCN CRYPTOGRAPHIC REQUIREMENTS

2.1 OBJECTIVE

35. This chapter defines the security requirements directly related to the correct implementation of the cryptographic mechanism supported by the TOE. The security requirements have been defined by CCN in conjunction with those specified in the [SOGIS-HEP] document by the SOG-IS participants.

36. Therefore, this section provides vendors with the cryptographic requirements that shall be met by the TOE in order to perform a valid implementation of the cryptographic mechanisms. In addition, it provides testers with the evaluation tasks necessary to verify the correct compliance with the requirements during the cryptographic mechanisms' evaluation process.

2.2 DEFINITIONS

Cryptographic Mechanisms

37. The cryptographic mechanisms can be classified according to the following main criteria:

a) Key Type:

- Symmetric mechanisms: cryptographic mechanisms that employ the same key for encryption and decryption operations. Therefore, this key must be secret and a key agreement protocol or a secure channel must be used to distribute it between the sender and receiver.
- Asymmetric mechanisms: cryptographic mechanisms that use public-private key pairs to perform cryptographic operations.
- Message digest functions: cryptographic mechanisms that do not require the use of keys. Their functionality is to provide a fixed-length digest from a variable-length input.

b) Security Strength: the security strength is usually expressed according to the attack complexity required to break the mechanism; therefore, an attack complexity of *n* bits means that it takes 2^n operations to perform the attack. The following table summarizes the cryptographic mechanisms that provide equivalent security strength.

Security Strength (Bits)	Symmetric Mechanisms Key Length (in bits)	Asymmetric Mechanisms	
		DSA, RSA, and Diffie-Hellman modulus length (in bits)	Elliptic Curves order of base point G (in bits)
112	112	2048	224

Security Strength (Bits)	Symmetric Mechanisms Key Length (in bits)	Asymmetric Mechanisms	
		DSA, RSA, and Diffie-Hellman modulus length (in bits)	Elliptic Curves order of base point G (in bits)
128	128	3072	256
192	192	7680	384
256	256	15360	512

Table 1. Security Strength of Keyed Cryptographic Mechanisms

Primitives and Constructions

38. It is common to distinguish cryptographic mechanisms according to their complexity. Cryptographic mechanisms are often built from more primary mechanisms, denoted as *cryptographic primitives*, in order to achieve higher-level security objectives. When this is the case, they are denoted as *cryptographic constructions*. The construction of cryptographic mechanisms upon an established cryptographic primitive(s) allows one to derive confidence from the prior analysis of the primitive(s): typically, the security of the cryptographic construction can be expressed in terms of the security of the underlying primitive(s), possibly together with some quantifiable bounded loss of assurance.

Self-Test Known Answer Tests (KATs)

39. Self-Test Known Answer Tests (KATs) are basic functional tests of the cryptographic mechanisms implemented in the TOE based on calculating the results for precalculated known inputs. These KATs are automatically executed by the TOE under certain conditions in order to verify that the implemented cryptographic mechanisms are working properly before their execution. In most cases, the self-test KATs are executed to produce valid results from known inputs. However, there are some cases in which is necessary to test invalid results.

Sensitive Data

40. Sensitive data refers to any data that shall be protected by the TOE in the sense of confidentiality and/or integrity in order to prevent disclosure to an attacker who might use advanced techniques such as side-channel attacks. Specifically, sensitive data encompass any cryptographic parameter whose disclosure could compromise the operation of any cryptographic mechanism, allowing an attacker to derive the encryption key, input, or outputs of the cryptographic mechanism.

41. Some examples of sensitive data are:

- Sensitive Security Parameters (SSPs): they are divided into critical security parameters (CSP) and public security parameters (PSP).
- Critical Security Parameters (CSPs): security-related information whose disclosure or modification can compromise the security of a TOE (e.g., secret and private cryptographic keys, authentication data such as passwords, PINs, etc.).
- Public Security Parameters (PSPs): security-related public information whose modification can compromise the security of the TOE (e.g., public cryptographic keys, public key certificates, self-signed certificates, etc.).
- Internal States of Cryptographic Primitives: they are related to the internal operations performed by some cryptographic mechanisms, for example, block ciphers and hashing cryptographic mechanisms.

2.3 CRYPTOGRAPHIC EVALUATION TASKS

42. This section contains the cryptographic requirements established by CCN that shall be considered by the vendors in the case of implementing a TOE with cryptographic capabilities.

43. In particular, the list of cryptographic requirements has been grouped into the following main topics:

- **Implementation of the Cryptographic Mechanisms** (constructions and primitives) to ensure that they correspond to those declared by the vendor.
- **Secure Management of SSPs** during their life cycle, from generation to destruction, employing approved cryptographic mechanisms for generation, entry/output, storage, and zeroization.
- **Implementation of Self-Tests** for the cryptographic mechanisms implemented by the TOE to ensure their correct operativity.
- **Implementation of Security Protections** to ensure mitigation of other attacks based on side-channel and differential fault analysis (DFA).

44. Therefore, the following list of cryptographic evaluation tasks related to the cryptographic implementation requirements established by CCN and comprising all the topics described above have been defined:

CRYPTOGRAPHIC EVALUATION TASK	CCN-MECHANISM	CERTIFICATION LEVEL
The **tester shall** verify that all the cryptographic mechanisms implemented by the TOE match with those declared by the vendor in **I1**.		**CL2 and CL3**

CRYPTOGRAPHIC EVALUATION TASK	CCN-SSP	CERTIFICATION LEVEL
The **tester shall** verify that the TOE performs secure management of Security Sensitive Parameters (SSPs) during their lifecycle, from generation to destruction, using approved cryptographic mechanisms for generation, entry/output, storage, and zeroization.		**CL2 and CL3**

CRYPTOGRAPHIC EVALUATION TASK	CCN-SELFTEST	CERTIFICATION LEVEL
The **tester shall** verify that the TOE correctly implements and executes the self-tests of the implemented cryptographic mechanisms.		**CL2 and CL3**

CRYPTOGRAPHIC EVALUATION TASK	CCN-MITIGATION	CERTIFICATION LEVEL
The **tester shall** verify that the TOE correctly implements security protections for the mitigation of other attacks based on side-channel attacks and differential fault analysis (DFA).		**CL2 and CL3**

45. For the correct completion of the evaluation tasks, each one has been divided into a set of cryptographic evaluation tests. The **tester shall** perform the associated mandatory tests, and also the implementation-dependent tests, if applicable, according to the TOE implementation.

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
CCN-MECHANISM Cryptographic Mechanisms	CCN-MECHANISM/BlockCipher	Impl-Dep
	CCN-MECHANISM/StreamCipher	Impl-Dep
	CCN-MECHANISM/Hash	Impl-Dep
	CCN-MECHANISM/SecretSharing	Impl-Dep
	CCN-MECHANISM/XOF	Impl-Dep
	CCN-MECHANISM/SymEncryption	Impl-Dep

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
	CCN-MECHANISM/DiskEncryption	Impl-Dep
	CCN-MECHANISM/MAC	Impl-Dep
	CCN-MECHANISM/EntityAuth	Impl-Dep
	CCN-MECHANISM/AuthEncryption	Impl-Dep
	CCN-MECHANISM/KeyProtection	Impl-Dep
	CCN-MECHANISM/KeyDerivation	Impl-Dep
	CCN-MECHANISM/PasswordMechanisms	Impl-Dep
	CCN-MECHANISM/RSA	Impl-Dep
	CCN-MECHANISM/FFDLOG	Impl-Dep
	CCN-MECHANISM/ECDLOG	Impl-Dep
	CCN-MECHANISM/AsymEncryption	Impl-Dep
	CCN-MECHANISM/DigitalSignature	Impl-Dep
	CCN-MECHANISM/KeyEstablishment	Impl-Dep
	CCN-MECHANISM/TRNG	Impl-Dep
	CCN-MECHANISM/DRNG	Impl-Dep
CCN-SSP SSP Management	CCN-SSP/Generation	Impl-Dep
	CCN-SSP/Transport	Impl-Dep
	CCN-SSP/Storage	Mandatory
	CCN-SSP/Zeroization.1	Mandatory
	CCN-SSP/Zeroization.2	Mandatory
CCN-SELFTEST Self-Tests	CCN-SELFTEST/Implementation	Mandatory

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
CCN-MITIGATION **Mitigation of Other Attacks**	CCN-MITIGATION/Mitigation	Mandatory

Table 2. Cryptographic Evaluation Tests defined for CCN Requirements

2.3.1 CRYPTOGRAPHIC EVALUATION TESTS DEFINITION

46. This subsection defines the cryptographic evaluation tests that shall be performed by the tester to accomplish the evaluation tasks, based on the cryptographic implementation requirements established by CCN.

2.3.1.1 CRYPTOGRAPHIC MECHANISMS IMPLEMENTATION

47. This first topic defines the cryptographic evaluation tests to verify the correct implementation by the TOE of each cryptographic mechanism declared by the vendor in **I1**, in order to fulfill the **CCN-MECHANISM** evaluation task.

2.3.1.1.1 Symmetric Elementary Primitives

2.3.1.1.1.1 Block Cipher

48. In case the TOE implements cryptographic primitives based on approved block ciphers:

- The **TOE shall:**
  - Include a self-test KAT for each block cipher cryptographic primitive implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each key length (128, 192, and 256 bits) implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each direction (ciphering and deciphering) implemented in the TOE, even if they are not in use.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/BlockCipher	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the block cipher cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**.	**I1** **I2** **I6**

CCN-MECHANISM/BlockCipher	Inputs
In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	

2.3.1.1.1.2 Stream Cipher

49. In case the TOE implements cryptographic primitives based on approved stream ciphers:

- The **TOE shall**:
  - Include a self-test KAT for each stream cipher cryptographic primitive implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each direction (ciphering and deciphering) implemented in the TOE, even if they are not in use.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/StreamCipher	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the stream cipher cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.1.3 Hash Functions

50. In case the TOE implements cryptographic primitives based on approved hash functions:

- The **TOE shall** include a self-test KAT for each hash primitive implemented in the TOE, even if they are not in use.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/Hash	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the hash cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.1.4 Secret Sharing

51. In case the TOE implements approved secret-sharing cryptographic primitives:

- The **TOE shall** include a self-test KAT for the shared secret recombination and generation operations.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/SecretSharing	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the secret-sharing cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.1.5 Extendable-Output Functions (XOF)

52. In case the TOE implements cryptographic primitives based on approved extendable-output functions (XOF):

- The **TOE shall** include a self-test KAT for each XOF primitive implemented in the TOE, even if they are not in use.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/XOF	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the XOF cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2 Symmetric Constructions

2.3.1.1.2.1 Symmetric Encryption (Confidentiality Only)

53. In case the TOE implements symmetric encryption based on approved block cipher constructions:

- The **TOE shall:**
  - Include a self-test KAT for each symmetric encryption construction implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each key length (128, 192, and 256 bits) implemented in the TOE, even if they are not in use.

- Include a self-test KAT for each direction (ciphering and deciphering) implemented in the TOE, even if they are not in use.

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/SymEncryption	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the symmetric encryption cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2.2 Disk Encryption

54. In case the TOE implements symmetric disk encryption based on approved block cipher constructions:

- The **TOE shall:**
  - Include a self-test KAT for each disk encryption construction implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each key length (256 and 512 bits for XTS operation mode) implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each direction (ciphering and deciphering) implemented in the TOE, even if they are not in use.

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/DiskEncryption	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the disk encryption cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2.3 Integrity Modes: Message Authentication Code

55. In case the TOE implements integrity modes based on approved message authentication code constructions:

- The **TOE shall:**
  - Include a self-test KAT (with correct and incorrect MAC values) for each MAC construction implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:
    - For MAC constructions relying on AES constructions, see section 2.3.1.1.2.1 "Symmetric Encryption (Confidentiality Only)".
    - For MAC constructions relying on Hash primitives, see section 2.3.1.1.1.3 "Hash Functions".
    - For the MAC constructions relying on XOF primitives, see section 2.3.1.1.1.5 "Extendable-Output Functions (XOF)".
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/MAC	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the MAC cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2.4 Symmetric Entity Authentication Schemes

56. In case the TOE implements approved symmetric entity authentication constructions to prove the identity against a verifier by demonstrating knowledge of a certain secret in a random challenge-response protocol:

- The **TOE shall**:
  - Include a self-test KAT for each symmetric entity authentication construction implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:
    - For symmetric entity authentication constructions relying on AES constructions, see section 2.3.1.1.2.1 "Symmetric Encryption (Confidentiality Only)".
    - For symmetric entity authentication constructions relying on MAC constructions, see section 2.3.1.1.2.3 "Integrity Modes: Message Authentication Code".
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/EntityAuth	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the symmetric entity authenticated constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2.5 Authenticated Encryption (AE)

57. In case the TOE implements approved authenticated encryption cryptographic constructions to provide confidentiality and integrity:

- The **TOE shall**:
  - Include a self-test KAT for each authenticated encryption construction implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each key length (128, 192, and 256 bits) implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each direction (ciphering and deciphering) implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:
    - For AE constructions relying on AES constructions, see section 2.3.1.1.2.1 "Symmetric Encryption (Confidentiality Only)".
    - For AE constructions relying on MAC constructions, see section 2.3.1.1.2.3 "Integrity Modes: Message Authentication Code".
    - For AE constructions relying on Stream cipher primitives, see section 2.3.1.1.1.2 "Stream Cipher".
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/AuthEncryption	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the authenticated encryption cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2.6 Key Protection

58. In case the TOE implements approved key protection constructions to secure store and transmit CSPs:

- The **TOE shall**:
  - Include a self-test KAT for each key protection construction implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each key length implemented in the TOE, even if they are not in use.
  - Include a self-test KAT for each direction (ciphering and deciphering) implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:
    - For key protection constructions relying on AES constructions, see section 2.3.1.1.2.1 "Symmetric Encryption (Confidentiality Only)".
    - For key protection constructions relying on MAC constructions, see section 2.3.1.1.2.3 "Integrity Modes: Message Authentication Code".
  - In the case of implementing an authenticated encryption cryptographic construction for key protection functionality, include the self-test KATs described in section 2.3.1.1.2.5 "Authenticated Encryption (AE)".
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/KeyProtection	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the key protection cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2.7 Key Derivation Function (KDF)

59. In case the TOE implements approved key derivation functions to derive cryptographic keys:

- The **TOE shall**:
  - Include a self-test KAT for each key derivation function implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:
    - For KDF constructions relying on Hash primitives, see section 2.3.1.1.1.3 "Hash Functions".

- For KDF constructions relying on MAC constructions, see section 2.3.1.1.2.3 "Integrity Modes: Message Authentication Code".
- For KDF constructions relying on other KDF constructions, the ones described in this section.

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/KeyDerivation	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the key derivation cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.2.8 Password Protection/Hashing Mechanisms

60. In case the TOE implements approved password protection/hashing constructions to protect/hash cryptographic keys:

- The **TOE shall**:
  - Include a self-test KAT for each password protection/hashing construction implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:
    - For password protection/hashing constructions relying on Hash primitives, see section 2.3.1.1.1.3 "Hash Functions".
    - For password protection/hashing constructions relying on MAC constructions, see section 2.3.1.1.2.3 "Integrity Modes: Message Authentication Code".

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/PasswordMechanisms	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the password protection/hashing cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.3 Asymmetric Elementary Primitives

2.3.1.1.3.1 RSA/Integer Factorization

61. In case the TOE implements approved cryptographic primitives based on the integer factorization problem (RSA):

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/RSA	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the RSA cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.3.2 Multiplicative Discrete Logarithm Problem

62. In case the TOE implements approved cryptographic primitives based on the multiplicative discrete logarithm problem (FFDLOG):

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/FFDLOG	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the FFDLOG cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.3.3 Additive Discrete Logarithm Problem

63. In case the TOE implements approved cryptographic primitives based on the additive discrete logarithm problem (ECDLOG):

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/ECDLOG	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the ECDLOG cryptographic primitives implemented by the TOE match with those declared by the vendor in **I1**.	**I1** **I2** **I6**

CCN-MECHANISM/ECDLOG	Inputs
In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	

2.3.1.1.4 Asymmetric Constructions

2.3.1.1.4.1 Asymmetric Encryption Schemes

64. In case the TOE implements asymmetric encryption constructions to provide data confidentiality based on approved RSA schemes:

- The **TOE shall**:
  - Include a self-test KAT (with correct and incorrect operations) for each RSA encryption scheme and modulus length implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:
    - For RSA encryption schemes relying on Hash primitives, see section 2.3.1.1.1.3 "Hash Functions".
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/AsymEncryption	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the asymmetric encryption cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.4.2 Digital Signature

65. In case the TOE implements digital signature constructions to provide data authentication based on approved asymmetric primitives:

- The **TOE shall**:
  - Include a self-test KAT for each digital signature construction implemented in the TOE:
    - For digital signature constructions based on the RSA asymmetric primitive, a self-test KAT for each RSA scheme and each modulus length implemented in the TOE, even if they are not in use.

  - For digital signature constructions based on the FFDLOG asymmetric primitive, a self-test KAT for each finite field group implemented in the TOE, even if they are not in use.
  - For digital signature constructions based on the ECDLOG asymmetric primitive, a self-test KAT for each elliptic curve implemented in the TOE, even if they are not in use.
- Include a self-test KAT for each operation (generation and verification) implemented in the TOE, even if they are not in use. For the verification operation, include correct and incorrect signature values.
- Include the self-test KATs for the underlying mechanisms:
  - For digital signature constructions relying on Hash primitives, see section 2.3.1.1.1.3 "Hash Functions".
  - For digital signature constructions relying on XOF primitives, see section 2.3.1.1.1.5 "Extendable-Output Functions (XOF)".

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/DigitalSignature	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the digital signature cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.4.3 Key Establishment

66. In case the TOE implements key establishment constructions to perform key agreements based on approved asymmetric primitives:

- The **TOE shall**:
  - Include a self-test KAT for each key establishment construction implemented in the TOE:
    - For key establishment constructions based on the FFDLOG asymmetric primitive, a self-test KAT for each finite field group implemented in the TOE, even if they are not in use.
    - For key establishment constructions based on the ECDLOG asymmetric primitive, a self-test KAT for each elliptic curve implemented in the TOE, even if they are not in use.
  - Include the self-test KATs for the underlying mechanisms:

- For key establishment constructions relying on key derivation functions, see section 2.3.1.1.2.7 "Key Derivation Function (KDF)".

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/KeyEstablishment	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the key establishment cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.5 Random Number Generators

2.3.1.1.5.1 True Random Number Generators

67. In case the TOE implements approved true random number generators (TRNGs):

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/TRNG	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the true random number generators (TRNGs) implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.1.5.2 Deterministic Random Number Generator Constructions

68. In case the TOE implements approved deterministic random number generator constructions to generate random data sequences:

- The **TOE shall**:
  - Include a self-test KAT for each DRNG construction implemented in the TOE, even if they are not in use, that executes all DRNG functions (instantiate, generated, and reseed).
  - Include the self-test KATs for the underlying mechanisms:
    - For DRNG constructions relying on MAC constructions, see section 2.3.1.1.2.3 "Integrity Modes: Message Authentication Code".

- For DRNG constructions relying on Hash primitives, see section 2.3.1.1.1.3 "Hash Functions".
- For DRNG constructions relying on AES constructions, see section 2.3.1.1.2.1 "Symmetric Encryption (Confidentiality Only)".

- The **tester shall** perform the following cryptographic evaluation test:

CCN-MECHANISM/DRNG	Inputs
In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify that the DRNG cryptographic constructions implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.2 SENSITIVE SECURITY PARAMETER MANAGEMENT

69. This second topic defines the cryptographic evaluation tests to verify the correct implementation by the TOE of the SSP management process declared by the vendor in **I1**, in order to fulfill the **CCN-SSP** evaluation task.

70. The SSP management process encompasses the entire lifecycle of SSPs managed by the TOE, taking into account their secure generation, establishment, entry/output, storage, transport, usage, and zeroization.

71. The following table summarizes the list of CSPs related to each cryptographic mechanism detailed in the *"Sensitive Data"* subsections of the [SOGIS-HEP] document.

Cryptographic Mechanism	Cryptographic Mode/Scheme	Critical Security Parameter (CSP) Description
Stream Cipher Primitive	**ChaCha20**	Encryption or Decryption Key
Secret Sharing Primitive	**Shamir**	Shared secret obtained using Shamir's Scheme
Symmetric Encryption (Confidentiality Only) Construction	**AES-CBC**	Encryption or Decryption Key
	AES-CBC-CS	
	AES-CFB	
	AES-CTR	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Critical Security Parameter (CSP) Description
	AES-OFB	
Disk Encryption Construction	**AES-XTS**	Encryption or Decryption Key
MAC Construction	**AES-CMAC**	Authentication Key used by the MAC cryptographic construction
	AES-CBC-MAC	
	HMAC	
	AES-GMAC	
	KMAC	
	Poly1305	
Authenticated Encryption Construction	**Encrypt-then-MAC**	Encryption or Decryption Key used by the symmetric encryption construction and Authentication Key used by the MAC cryptographic construction
	CCM	Authenticated Encryption or Decryption Key
	GCM	
	EAX	
	ChaCha20-Poly1305	Encryption or Decryption Key used by the ChaCha20 stream cipher primitive and Authentication Key used by the Poly1305 cryptographic construction
Key Protection Construction	**SIV**	Key Encryption or Decryption Key
	AES-Keywrap	
Key Derivation Construction	**NIST SP800-56 A/B/C**	Input Secret to generate the new derived key
	NIST SP800-108	
	ANSI X9.63 KDF	
	PBKDF2	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Critical Security Parameter (CSP) Description
	HKDF	
Password Protection/Hashing Construction	**Argon2id**	Input Secret to perform password protection/hashing operations
	PBKDF2	
	SCRYPT	
Asymmetric Encryption Scheme	**RSA OAEP**	The private component of RSA key pairs in asymmetric decryption operations
	RSA PKCS#1v1.5	
Digital Signature Construction	**RSA PSS**	The private component of RSA key pairs in digital signature generation operations
	RSA PKCS#1v1.5	
	KCDSA	The private component of FFDLOG key pairs in digital signature generation operations
	Schnorr	
	DSA	
	EC-KCDSA	The private component of ECDLOG key pairs in digital signature generation operations
	EC-DSA	
	EC-GDSA	
	EC-Schnorr	
	EdDSA	
	XMSS	The private component of key pairs in digital signature generation operations
	ML-DSA	
	Falcon	
	SLH-DSA	
Key Establishment Scheme	**DH**	Shared Secret obtained as a result of the key establishment procedure The private component of key pairs in secret-sharing computation operations
	EC-DH	
	DLIES-KEM	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Critical Security Parameter (CSP) Description
	ECIES-KEM	
	ML-KEM	Shared Secret obtained as a result of the key encapsulation and decapsulation mechanisms The private component of key pairs (the decapsulation key) in key encapsulation computation operations
	FrodoKEM	
Deterministic Random Number Generator Construction	**HMAC-DRBG**	Seed used to initialize or re-seed the DRNG The internal state of DRNG generated in each generation operation
	Hash-DRBG	
	CTR-DRBG (AES)	

Table 3. CSP Definition for each Cryptographic Mechanism

72. The following table summarizes the list of PSPs related to each cryptographic mechanism detailed in the *"Sensitive Data"* subsections of the [SOGIS-HEP] document.

Cryptographic Mechanism	Cryptographic Mode/Scheme	Public Security Parameter (PSP) Description
Asymmetric Encryption Scheme	**RSA OAEP**	The public component of RSA key pairs in asymmetric encryption operations
	RSA PKCS#1v1.5	
Digital Signature Construction	**RSA PSS**	The public component of RSA key pairs in digital signature verification operations
	RSA PKCS#1v1.5	
	KCDSA	The public component of FFDLOG key pairs in digital signature verification operations
	Schnorr	
	DSA	
	EC-KCDSA	The public component of ECDLOG key pairs in digital signature verification operations
	EC-DSA	
	EC-GDSA	
	EC-Schnorr	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Public Security Parameter (PSP) Description
	EdDSA	
	XMSS	The public component of key pairs in digital signature verification operations
	ML-DSA	
	Falcon	
	SLH-DSA	
Key Establishment Scheme	**DH**	The public component of FFDLOG key pairs in secret-sharing computation operations
	EC-DH	The public component of ECDLOG key pairs in secret sharing computation operations
	DLIES-KEM	The public component of FFDLOG key pairs in secret-sharing computation operations
	ECIES-KEM	The public component of ECDLOG key pairs in secret sharing computation operations
	ML-KEM	The public component of key pairs (the encapsulation key) in key decapsulation computation operations
	FrodoKEM	

Table 4. PSP Definition for each Cryptographic Mechanism

2.3.1.2.1 Sensitive Security Parameter Generation and Establishment

73. In case the TOE implements sensitive security parameter generation:

- The **TOE shall**:
  - Implement one of the following approved cryptographic mechanisms to perform SSP generation:
    - An approved random number generator, according to the following configurations:
      - For symmetric mechanisms, an approved DRNG (see section 3.3.1.1.5.2 "Deterministic Random Number Generators").

- For asymmetric keys based on RSA, an approved RSA key pair generation scheme (see section 3.3.1.1.3.1 "RSA/Integer Factorization").

For asymmetric private keys based on FFDLOG, an approved random uniform modular distribution (see section 0 "

- Multiplicative Discrete Logarithm Problem").
- For asymmetric private keys based on ECDLOG, an approved random uniform modular distribution (see 3.3.1.1.3.3 "Additive Discrete Logarithm Problem").

- An approved key establishment construction (see section 3.3.1.1.4.3 "Key Establishment"). In this case:
  - Implement an approved key derivation function establishment as a postprocessing method (see section 3.3.1.1.2.7 "Key Derivation Functions").
  - For any key establishment construction that relies on pre-existing secrets (e.g., ephemeral components in Diffie-Hellman or master secret in TLS Record Protocol), employ at least 125 bits of entropy for the generation of pre-existing secrets.
- An approved key derivation function (see section 3.3.1.1.2.7 "Key Derivation Functions").

- The **tester shall** perform the following cryptographic evaluation test:

CCN-SSP/Generation	**Inputs**
In the case of **CL2** evaluations, verify in **I1** that the SSP generation methods declared by the vendor are one of the following approved cryptographic mechanisms: ▪ An approved random number generator, in this case, verify that it complies with the following configurations: - For symmetric mechanisms, an approved DRNG. - For asymmetric keys based on RSA, an approved RSA key pair generation scheme. - For asymmetric private keys based on FFDLOG, an approved random uniform modular distribution. - For asymmetric private keys based on ECDLOG, an approved random uniform modular distribution. ▪ An approved key establishment construction, in this case, verify that: - The key establishment construction declared by the vendor uses an approved key derivation function as a postprocessing method.	**I1** **I2** **I6**

CCN-SSP/Generation	Inputs
- The generation of any pre-existing secret in SSP establishment is performed using at least 125 bits of entropy. ▪ An approved key derivation function. In addition, operate the TOE (using **I2**) to verify that: - The SSP generation methods implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, use **I1**, operate the TOE (using **I2**), and perform a source code review (using **I6**) to verify the above-mentioned.	

2.3.1.2.2 Sensitive Security Parameter Entry/Output

74. In case the TOE implements sensitive security parameter entry/output:

- The **TOE shall**:
  - Implement a secure key distribution channel ensuring the authenticity and integrity of the PSPs using approved cryptographic mechanisms.
  - Implement a secure key distribution channel ensuring the authenticity, integrity, and confidentiality of the CSPs using approved key protection mechanisms.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-SSP/Transport	Inputs
In the case of **CL2** evaluations, verify in **I1** that: - The PSP entry/output is protected in authenticity and integrity using an approved cryptographic mechanism. - The CSP entry/output is protected in authenticity, integrity, and confidentiality using an approved key protection mechanism. In addition, operate the TOE (using **I2**) to verify that: - The SSP transport methods implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, use **I1**, operate the TOE (using **I2**), and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.2.3 Sensitive Security Parameter Storage

75. Regarding the storage of sensitive security parameters:

- The **TOE shall**:
  - Ensure the authenticity and integrity of the stored PSPs to protect them from unauthorized modifications or substitutions using approved cryptographic mechanisms.
  - Ensure the confidentiality, authenticity, and integrity of the stored CSPs to protect them from disclosure using approved key protection mechanisms.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-SSP/Storage	Inputs
In the case of **CL2** evaluations, verify in **I1** that: - The PSP storage is protected in authenticity and integrity using an approved cryptographic mechanism. - The CSP storage is protected in authenticity, integrity, and confidentiality using an approved key protection mechanism. In addition, operate the TOE (using **I2**) to verify that: - The SSP storage methods implemented by the TOE match with those declared by the vendor in **I1**. In the case of **CL3** evaluations, use **I1**, operate the TOE (using **I2**), and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

2.3.1.2.4 Sensitive Security Parameter Zeroization

76. Regarding the zeroization of sensitive security parameters, it is necessary to perform the zeroization of critical security parameters (CSP) stored in volatile and non-volatile TOE memory, therefore:

- The **TOE shall**:
  - Implement a zeroization mechanism to destroy all stored CSPs automatically without operator intervention once they are no longer needed, making it impossible to recover them after the zeroization. The only approved zeroization procedure is to overwrite CSPs with zeros, ones, or random values. Therefore, overwriting any CSP with another CSP is not considered a valid zeroization mechanism.

  Note: For unprotected CSPs stored in non-volatile memory, the term “zeroize once they are no longer needed” means “zeroize after the usage of the CSP”.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-SSP/Zeroization.1	Inputs
In the case of **CL2** evaluations, verify that the list of CSPs declared by the vendor in **I1** encompasses all the CSPs related to each implemented cryptographic mechanism as detailed in Table 3 "CSP Definition for each Cryptographic Mechanism". In addition, operate the TOE (using **I2**) and review the evidence provided in **I1** to verify that: - All the CSPs are zeroized using only approved zeroization mechanisms and without any operator intervention once they are no longer needed. - No CSP is being directly overwritten by another CSP. - It is not possible to recover any zeroized CSP. In the case of **CL3** evaluations, operate the TOE (using **I2**), review the evidence provided in **I1**, and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

77. Moreover, it is necessary to perform the zeroization of the temporally sensitive data (temporally stored CSPs, key components, and ephemeral CSPs) used in the operation of the different implemented cryptographic mechanisms once they are no longer needed, therefore:

- The **TOE shall**:
  - In the case of **CL2** evaluations, overwrite sensitive data with new data or use an approved zeroization procedure to overwrite them with zeros, ones, or random values.
  - In the case of **CL3** evaluations, zeroize sensitive data using an approved zeroization procedure to overwrite temporally sensitive data with zeros, ones, or random values. Moreover, sensitive data shall not be stored even partially, in volatile or non-volatile memory.

78. Temporally sensitive data are those obtained from the execution of a cryptographic mechanism, as detailed in the *"Sensitive Data"* subsections in the [SOGIS-HEP] document, except the CSPs listed in Table 3 "CSP Definition for each Cryptographic Mechanism". The following table summarizes the list of temporally sensitive data related to each cryptographic mechanism:

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
Block Cipher Primitive	**AES**	- Each round key in the AES algorithm execution - Erroneous ciphertexts - Plaintext on the encryption side

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
		- Temporally variables such as the state and counter of each round - Initialization vectors on the encryption side - Counters on the encryption side
Stream Cipher Primitive	**ChaCha20**	- Internal states of each round - The output of each round - Plaintext - Nonce - Erroneous ciphertexts - During the encryption, the inputs of the XOR operation - During the decryption, the input and the output of the XOR operation
Hash Primitive	**SHA-2 Family** **SHA-3 Family** **BLAKE2b**	- The input of the hash function - Internal states during the execution because they could be used to derive the input or the output of the function
Secret Sharing Primitive	**Shamir**	Each of the parts (specific to each user) that are needed to generate the shared secret
XOF	**SHAKE** **cSHAKE**	- The input of the XOF - Internal states during the execution because they could be used to derive the input or the output of the function
Symmetric Encryption (Confidentiality Only) Construction	**AES-CBC**	Sensitive data related to the AES block cipher primitive shall be zeroized No additional sensitive data related to this scheme
	AES-CBC-CS	Sensitive data related to the AES block cipher primitive shall be zeroized No additional sensitive data related to this scheme
	AES-CFB	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - Considering that CFB is a stream mode, each output of the block cipher is sensitive data because it allows to derive the plaintext - During the encryption, the inputs of the XOR operation

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
		- During the decryption, one input (the key stream) and the output of the XOR operation
	AES-CTR	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - Considering that CTR is a stream mode, each output of the block cipher is sensitive data because it allows to derive the plaintext - During the encryption, the inputs of the XOR operation - During the decryption, one input (the key stream) and the output of the XOR operation
	AES-OFB	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - Considering that OFB is a stream mode, each output of the block cipher is sensitive data because it allows to derive the plaintext - During the encryption, the inputs of the XOR operation - During the decryption, one input (the key stream) and the output of the XOR operation
Disk Encryption Construction	**AES-XTS**	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - Encrypted tweak values
MAC Construction	**AES-CMAC**	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - The input message and the generated MAC - Key k_0 derived from the initial key by encrypting a constant message with the initial key - Keys k_1 and k_2 derived from k_0 using shift and XOR operations that are used to generate the MAC with intermediate values

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
		- The input and output of shifts and XOR operations
	AES-CBC-MAC	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - The input message and the generated MAC
	HMAC	Sensitive data related to the hash primitive shall be zeroized Additional sensitive data related to this scheme: - The input message and the generated MAC - Results of the XOR operations internally performed
	AES-GMAC	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - The input message and the generated MAC - The secret H derived using the authentication key computed as $E_k(0^{128})$
	KMAC	Sensitive data related to the cSHAKE primitive shall be zeroized Additional sensitive data related to this scheme: - The input message and the generated MAC - Internal states during the execution
	Poly1305	- The input message and the generated MAC
Authenticated Encryption Construction	**Encrypt-then-MAC**	Sensitive data related to the underlying AES block cipher primitive and the underlying MAC cryptographic construction shall be zeroized No additional sensitive data related to this scheme
	AES-CCM	Sensitive data related to the AES CTR and the CBC-MAC cryptographic constructions shall be zeroized No additional sensitive data related to this scheme

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
	AES-GCM	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme: - The secret H derived using the authentication key computed as $E_k(0^{128})$ - The output of the block cipher because it allows to derive the plaintext - During the encryption, the inputs of the XOR operation - During the decryption, one input (the key stream) and the output of the XOR operation
	AES-EAX	Sensitive data related to the AES CTR and the CMAC cryptographic constructions shall be zeroized No additional sensitive data related to this scheme
	ChaCha20-Poly1305	Sensitive data related to the ChaCha20 stream primitive and Poly1305 construction shall be zeroized No additional sensitive data related to this scheme
Key Protection Construction	**AES-SIV**	Sensitive data related to the AES CTR and the CMAC cryptographic constructions shall be zeroized No additional sensitive data related to this scheme
	AES-Keywrap	Sensitive data related to the AES block cipher primitive shall be zeroized No additional sensitive data related to this scheme

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
Key Derivation Construction	**NIST SP800-56 A/B/C OneStep**	If relies on a hash function, sensitive data related to the hash primitive shall be zeroized If relies on an HMAC function, sensitive data related to the HMAC construction shall be zeroized If relies on a KMAC function, sensitive data related to the KMAC construction shall be zeroized No additional sensitive data related to this scheme
	NIST SP800-56 A/B/C TwoStep	If relies on an HMAC function, sensitive data related to the HMAC construction shall be zeroized If relies on a CMAC function, sensitive data related to the CMAC construction shall be zeroized Sensitive data related to the NIST SP800-108 KDF cryptographic construction shall be zeroized No additional sensitive data related to this scheme
	NIST SP800-108	If relies on an HMAC function, sensitive data related to the HMAC construction shall be zeroized If relies on a CMAC function, sensitive data related to the CMAC construction shall be zeroized If relies on a KMAC function, sensitive data related to the KMAC construction shall be zeroized Additional sensitive data related to this scheme: - The temporary KDKs used to derivate the final key - The intermediate results of the iterations
	ANSI X9.63 KDF	If relies on a hash function, sensitive data related to the hash primitive shall be zeroized No additional sensitive data related to this scheme

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
	PBKDF2	Sensitive data related to the HMAC construction shall be zeroized
	HKDF	Sensitive data related to the HMAC construction shall be zeroized Additional sensitive data related to this scheme: - The pseudo-random key computed in the "extract" phase
Password Protection/ Hashing Mechanisms	**Argon2id**	Sensitive data related to the BLAKE2b primitive shall be zeroized Additional sensitive data related to this scheme: - The derived password when used for password verification
	PBKDF2	Sensitive data related to the HMAC construction shall be zeroized Additional sensitive data related to this scheme: - The derived password when used for password verification
	SCRYPT	Sensitive data related to the PBKDF2 construction shall be zeroized Additional sensitive data related to this scheme: - All the blocks obtained through PBKDF2 outputs, as well as intermediate states in the computations that involve those blocks
Asymmetric Primitive	**RSA**	- The result of the exponentiation - Temporary variables within the exponentiation - When using the CRT mode, the modulus, and possible erroneous outputs - Other erroneous outputs

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
	FFDLOG	- The temporary variables within the exponentiation - The output in exponentiations using unintended subgroups - Erroneous computed results during modular exponentiations - Other erroneous outputs In addition, depending on the mechanism used to perform simple or multi-exponentiation, the following parameter may be considered sensitive data: - The base(s) - The exponent(s) - The result
	ECDLOG	- The temporary variables within the multiplication - The output in multiplications using unintended curves or corrupted parameters - Erroneous computed results during EC scalar multiplications - Other erroneous outputs In addition, depending on the mechanism used to perform the simple or EC multi-scalar multiplication, the following parameter may be considered sensitive data: - The base(s) - The scalar(s) - The resulting point
Asymmetric Encryption Scheme	**RSA-OAEP**	Sensitive data related to the RSA asymmetric primitive and SHA hash primitive shall be zeroized Additional sensitive data related to this scheme: - The plaintext to be encrypted in the encryption operation - The ciphertext (encoded plaintext) in the case of decryption operations - Output of the hash function employed - Intermediate values for OAEP encoding and decoding operations

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
	RSA-PKCS#1v1.5	Sensitive data related to the RSA asymmetric primitive shall be zeroized Additional sensitive data related to this scheme: - The plaintext to be encrypted in the encryption operation - The ciphertext (encoded plaintext) in the case of decryption operations
Digital Signature Construction	**RSA-PSS** **RSA-PKCS#1v1.5**	Sensitive data related to the RSA asymmetric primitive and SHA hash primitive shall be zeroized Additional sensitive data related to this scheme: - The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that case, it is a sensitive parameter as well as its digest and the signature
	DSA **KCDSA** **Schnorr**	Sensitive data related to the FFDLOG asymmetric primitive and SHA hash primitive shall be zeroized Additional sensitive data related to this scheme: - The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that case, it is a sensitive parameter as well as its digest and the signature - If the message and/or signature are considered sensitive, then all the intermediary results for signature generation and verification are considered sensitive - During the signature generation, the random per-message value (k) is considered sensitive data because it is the exponent used in the modular exponentiation - All the intermediary results for the computation of the signature(s) where the private key and the per-message value are involved

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
	EC-DSA **EC-KCDSA** **EC-GDSA** **EC-Schnorr**	Sensitive data related to the ECDLOG asymmetric primitive and SHA hash primitive shall be zeroized (if applicable) Additional sensitive data related to this scheme: - The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that case, it is a sensitive parameter as well as its digest and the signature - If the message and/or signature are considered sensitive, then all the intermediary results for signature generation and verification are considered sensitive - During the signature generation, the random per-message value (k) is considered sensitive data because it is a scalar in EC scalar multiplication - All the intermediary results for the computation of the signature(s) where the private key and the per-message value are involved
	EdDSA	Sensitive data related to the ECDLOG asymmetric primitive, SHA hash primitive, and XOF primitive shall be zeroized (if applicable) Additional sensitive data related to this scheme: - The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that case, it is a sensitive parameter as well as its digest and the signature - If the message and/or signature are considered sensitive, then all the intermediary results for signature generation and verification are considered sensitive - All the intermediary value that uses the output of the private key hash - All the intermediary results for the computation of the signature(s) where the private key and the per-message value are involved

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
	XMSS	Sensitive data related to the SHA hash and XOF primitives shall be zeroized Additional sensitive data related to this scheme: - The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that case, it is a sensitive parameter as well as its digest and the signature - If the message and/or signature are considered sensitive, then all the intermediary results for signature generation and verification are considered sensitive - All the intermediary results for the computation of the signature(s) where the private key and the per-message value are involved - All the intermediate result values in the *WOTS+* algorithm (such as the message in base *w* or the checksum) - All sensitive parameters of the pseudo-random function
	ML-DSA **Falcon**	Sensitive data related to the underlying mechanisms shall be zeroized Additional sensitive data related to this scheme: - The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that case, it is a sensitive parameter as well as its digest and the signature - If the message and/or signature are considered sensitive, then all the intermediary results for signature generation and verification are considered sensitive - All the intermediary results for the computation of the signature(s) where the private key and the per-message value are involved
	SLH-DSA	Sensitive data related to the underlying mechanisms shall be zeroized Additional sensitive data related to this scheme:

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
		- The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that case, it is a sensitive parameter as well as its digest and the signature - If the message and/or signature are considered sensitive, then all the intermediary results for signature generation and verification are considered sensitive - All the intermediary results for the computation of the signature(s) where the private key and the per-message value are involved - The hypertree node keys and the *WOTS+* private keys
Key Establishment Scheme	**DH**	Sensitive data related to the FFDLOG asymmetric primitive and the KDF construction shall be zeroized Additional sensitive data related to this scheme: - The random secret values used as the exponent in DH to produce the shared secret and all the data derived from the base secret
	EC-DH	Sensitive data related to the ECDLOG asymmetric primitive and the KDF construction shall be zeroized Additional sensitive data related to this scheme: - The random secret values used as the scalar in ECDH to produce the shared secret and all the data derived from the base secret
	DLIES-KEM	Sensitive data related to the FFDLOG asymmetric primitive and the KDF construction shall be zeroized Additional sensitive data related to this scheme: - The random secret values used as the exponent in the DLIES-KEM key generation operation to produce the shared secret - The random secret value applied to the other party's public key as the exponent in the DLIES-KEM encapsulation operation to produce the shared secret is considered

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
		sensitive data. Moreover, the data derived during the operation is considered sensitive data
	ECIES-KEM	Sensitive data related to the ECDLOG asymmetric primitive and the KDF construction shall be zeroized Additional sensitive data related to this scheme: - The random secret values used as the scalar in the ECIES-KEM key generation operation to produce the shared secret - The random secret value applied to the other party's public key as the scalar in the ECIES-KEM encapsulation operation to produce the shared secret is considered sensitive data. Moreover, the data derived during the operation is considered sensitive data
	ML-KEM	Sensitive data related to the underlying mechanisms shall be zeroized No additional sensitive data related to this scheme
	FrodoKEM	Sensitive data related to the underlying mechanisms shall be zeroized No additional sensitive data related to this scheme
Deterministic Random Number Generator Construction	**HMAC-DRBG**	Sensitive data related to the HMAC construction shall be zeroized Additional sensitive data related to this scheme: - The random bit output by the DRNG mechanism is often considered sensitive data because they are usually used to generate secret keys
	Hash-DRBG	Sensitive data related to the hash function shall be zeroized Additional sensitive data related to this scheme: - The random bit output by the DRNG mechanism is often considered sensitive data because they are usually used to generate secret keys

Cryptographic Mechanism	Cryptographic Mode/ Scheme	Temporally Sensitive Data for CL2 and CL3
	CTR-DRBG	Sensitive data related to the AES block cipher primitive and AES-CTR construction shall be zeroized Additional sensitive data related to this scheme: - The random bit output by the DRNG mechanism is often considered sensitive data because they are usually used to generate secret keys

Table 5. Temporally Sensitive Data for each Cryptographic Mechanism

- The **tester shall** perform the following cryptographic evaluation test:

CCN-SSP/Zeroization.2	**Inputs**
In the case of **CL2** and **CL3** evaluations, verify that the list of temporally sensitive data declared by the vendor in **I1** encompasses all the temporally sensitive data related to each implemented cryptographic mechanism as detailed in Table 5 "Temporally Sensitive Data for each Cryptographic Mechanism". In the case of **CL2** evaluations, operate the TOE (using **I2**) and review the evidence provided in **I1** to verify that: - All temporally sensitive data are overwritten with new data or zeroized using only approved zeroization mechanisms and without any operator intervention once they are no longer needed. - It is not possible to recover any zeroized temporally sensitive data. In the case of **CL3** evaluations, operate the TOE (using **I2**), review the evidence provided in **I1**, and perform a source code review (using **I6**) to verify that: - All temporally sensitive data are zeroized using only approved zeroization mechanisms and without any operator intervention once they are no longer needed. - It is not possible to recover any zeroized temporally sensitive data.	**I1** **I2** **I6**

2.3.1.3 CRYPTOGRAPHIC MECHANISMS SELF-TEST

79. This third topic defines the cryptographic evaluation test to verify the correct implementation by the TOE of the cryptographic mechanisms self-tests declared by the vendor in **I1**, in order to fulfill the **CCN-SELFTEST** evaluation task.

80. The cryptographic mechanisms self-tests are diagnostic procedures used to verify the correct operativity of the implemented cryptographic mechanisms.

81. Regarding the implementation of the cryptographic mechanisms self-tests:

- The **TOE shall** implement and execute the self-tests associated with the implemented cryptographic mechanisms. These are defined throughout section 2.3.1.1 "Cryptographic Mechanisms Implementation".
- The **vendor shall** provide the self-test input vectors associated with the implemented cryptographic mechanisms (**I7**) to allow the tester to perform the self-test validation process using the CCN Cryptographic Tool.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-SELFTEST/Implementation	**Inputs**
In the case of **CL2** evaluations, operate the TOE (using **I2**), use the self-test input vectors (**I7**), and review the evidence provided in **I1** to verify that: - The parameterization used in the self-test KATs complies with the requirements defined in Section 3 "CCN Agreed Cryptographic Mechanisms". - The self-test KATs implemented for each cryptographic mechanism are valid. For this purpose, the CCN Cryptographic Tool shall be used as the cryptographic reference implementation. - The TOE correctly implements and executes all the cryptographic mechanisms self-tests. In the case of **CL3** evaluations, operate the TOE (using **I2**), use the self-test input vectors (**I7**), review the evidence provided in **I1**, and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6** **I7**

2.3.1.4 MITIGATION OF OTHER ATTACKS

82. This fourth topic defines the cryptographic evaluation test to verify the implementation by the TOE of countermeasures to mitigate other attacks declared by the vendor in **I1**, in order to fulfill the **CCN-MITIGATION** evaluation task.

83. Nowadays, products can be attacked without requiring the attacker to modify or access them. These attacks are known as side-channel attacks and they target the leakage of secret information through unintended channels in the implementation of a mechanism. These channels are related to physical effects such as power consumption, timing characteristics, and electromagnetic radiation.

84. This kind of attack can change the behavior of an integrated circuit (IC) in order to modify its normal operation by causing an exploitable error. This modification can occur by causing the IC to operate outside voltage, temperature, and clock frequency ranges or by applying more than one external power source during IC operation.

- Therefore, to protect the critical security parameters against side-channel attacks and differential fault analysis (DFA), the **TOE shall** implement countermeasures to mitigate these attacks.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-MITIGATION/Mitigation	**Inputs**
In the case of **CL2** evaluations, review the evidence provided in **I1** to verify that the TOE implements countermeasures to mitigate side-channel attacks and differential fault analysis (DFA). In the case of **CL3** evaluations, review the evidence provided in **I1** and perform a source code review (using **I6**) to verify the above-mentioned. Moreover, these countermeasures shall be tested according to the lab experience.	**I1** **I2** **I6**

3 CCN AGREED CRYPTOGRAPHIC MECHANISMS

3.1 OBJECTIVE

85. This chapter defines the cryptographic mechanisms and protocols that are considered as agreed upon by CCN according to the [CCN-STIC-221] guidance.

86. Therefore, it provides vendors with recommendations and additional considerations that shall be met by the TOE to perform a secure implementation of the cryptographic mechanisms and protocols. In addition, it provides testers with the evaluation tasks necessary to verify the correct compliance with the requirements during the cryptographic mechanisms' evaluation process.

3.2 DEFINITIONS

Cryptographic Mechanisms

87. Agreed cryptographic mechanisms are subdivided into two categories, according to their estimated robustness. For each category of agreed mechanisms, the principles governing the management of the terms of their validity are summarized below.

    a) **Recommended Mechanisms (R):** they fully reflect the current state-of-the-art in cryptographic security engineering. They currently offer a security level of at least 128 bits, are supported by strong security mechanisms, and can provide an adequate level of security against all known and suspected threats, even taking into account the generally anticipated increases in computing power. Residual threats to their security may arise from potential innovative developments.

    b) **Legacy Mechanisms (L):** they are deployed on a large scale, currently offer a security level of at least 100 bits, and are considered to provide acceptable security in the short term, but should be phased out as soon as possible because they no longer fully reflect the state-of-the-art and suffer from some security assurance limitations compared to the recommended mechanisms. Consequently, a validity period is defined for legacy mechanisms.

       For legacy mechanisms that are expected to become vulnerable shortly, the validity period is given by a deprecation deadline: the mechanism is no longer accepted after the given deadline. The notation L[2030] indicates that the acceptance of the mechanisms expires on December 31, 2030.

       For legacy mechanisms that do not suffer from these limitations, the given validity period should be interpreted as a minimum validity period, during which the mechanism is expected to remain acceptable. The notation L[2030+] indicates that the acceptance of the legacy mechanism goes at least until the end of 2030.

Primitives and Constructions

88. Cryptographic mechanisms are often built upon more elementary mechanisms, denoted as cryptographic primitives, to achieve higher-level security goals. When this is the case, we call them cryptographic constructions.

89. The construction of cryptographic mechanisms from one or more established cryptographic primitives allows to derive confidence from the prior analysis of the primitive(s): usually, the security of the cryptographic construction can be expressed in terms of the security of the underlying primitive(s), possibly together with some quantifiable and limited loss of assurance.

Note: For a construction to be considered agreed, only agreed primitives shall be used unless explicitly stated otherwise. Moreover, for a construction that is marked "recommended" to result in a recommended mechanism, the underlying primitive(s) shall also be recommended. If it is marked "recommended" but an underlying primitive is marked "legacy", the resulting mechanism is considered "legacy". It is also noted that specific notes related to a cryptographic primitive also apply to constructions involving that primitive, even if these notes are not repeated.

3.3 CRYPTOGRAPHIC EVALUATION TASKS

90. This section contains the cryptographic mechanisms and protocols agreed upon by CCN that shall be considered by the vendors in the case of implementing a TOE with cryptographic capabilities.

91. The following cryptographic evaluation tasks related to the cryptographic mechanisms and protocols agreed upon by CCN have been defined:

CRYPTOGRAPHIC EVALUATION TASK	CCN-AGREED	CERTIFICATION LEVEL
The **tester shall** verify that all the cryptographic mechanisms implemented by the TOE are considered as agreed according to this methodology and comply with the specific notes. - For **CL1**, agreed means legacy or recommended cryptographic mechanisms. - For **CL2** and **CL3**, agreed means only recommended cryptographic mechanisms. **Note:** In the case of **CL2** and **CL3** evaluations, all implemented cryptographic mechanisms shall be verified, even if they are not in use.		**CL1, CL2 and CL3**

CRYPTOGRAPHIC EVALUATION TASK	CCN-PROTOCOL	CERTIFICATION LEVEL
The **tester shall** verify that all the cryptographic protocols implemented by the TOE are considered as agreed according to this methodology and comply with the specific notes. - For **CL1**, agreed means legacy or recommended cryptographic protocols and configurations. - For **CL2** and **CL3**, agreed means only recommended cryptographic protocols and configurations. **Note:** In the case of **CL2** and **CL3** evaluations, all implemented cryptographic protocols shall be verified, even if they are not in use.		**CL1, CL2 and CL3**

92. For the correct completion of the evaluation tasks, each one of them has been divided into a set of cryptographic evaluation tests. The **tester shall** perform the associated implementation-dependent tests, if applicable, according to the TOE implementation.

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
CCN-AGREED **Agreed Cryptographic Mechanisms**	CCN-AGREED/BlockCipher	Impl-Dep
	CCN-AGREED/StreamCipher	Impl-Dep
	CCN-AGREED/Hash	Impl-Dep
	CCN-AGREED/SecretSharing	Impl-Dep
	CCN-AGREED/XOF	Impl-Dep
	CCN-AGREED/SymEncryption	Impl-Dep
	CCN-AGREED/DiskEncryption	Impl-Dep
	CCN-AGREED/MAC	Impl-Dep
	CCN-AGREED/EntityAuth	Impl-Dep
	CCN-AGREED/AuthEncryption	Impl-Dep

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
	CCN-AGREED/KeyProtection	Impl-Dep
	CCN-AGREED/KeyDerivation	Impl-Dep
	CCN-AGREED/PasswordMechanisms	Impl-Dep
	CCN-AGREED/RSA	Impl-Dep
	CCN-AGREED/RSAKeys	Impl-Dep
	CCN-AGREED/FFDLOG	Impl-Dep
	CCN-AGREED/FFDLOGKeys	Impl-Dep
	CCN-AGREED/ECDLOG	Impl-Dep
	CCN-AGREED/ECDLOGKeys	Impl-Dep
	CCN-AGREED/AsymEncryption	Impl-Dep
	CCN-AGREED/DigitalSignature	Impl-Dep
	CCN-AGREED/KeyEstablishment	Impl-Dep
	CCN-AGREED/TRNGEntropy	Impl-Dep
	CCN-AGREED/PTRNG	Impl-Dep
	CCN-AGREED/NPTRNG	Impl-Dep
	CCN-AGREED/DRNG	Impl-Dep
	CCN-AGREED/DRNGScheme	Impl-Dep
CCN-PROTOCOL Agreed Cryptographic Protocols	CCN-PROTOCOL/TLS	Impl-Dep
	CCN-PROTOCOL/SSH	Impl-Dep
	CCN-PROTOCOL/IPsec	Impl-Dep

Table 6. Cryptographic Evaluation Tests defined for CCN Agreed & Protocol

3.3.1 CRYPTOGRAPHIC EVALUATION TESTS DEFINITION

93. This subsection defines the cryptographic evaluation tests that shall be performed by the tester to accomplish the evaluation tasks, based on the agreed cryptographic mechanisms and protocols established by CCN.

3.3.1.1 CRYPTOGRAPHIC MECHANISMS IMPLEMENTATION

94. This first topic defines the cryptographic evaluation tests to verify that each cryptographic mechanism declared by the vendor in **I1** is considered agreed upon by CCN, in order to fulfill the **CCN-AGREED** evaluation task.

3.3.1.1.1 Symmetric Elementary Primitives

95. Symmetric primitives utilize the same cryptographic key for both encryption and decryption operations (except for hash and XOF functions which are keyless functions). For these primitives, the security consists of the confidentiality of the key shared by legitimate users.

3.3.1.1.1.1 Block Cipher

96. A block cipher is a symmetric elementary primitive based on a family of permutations of $\{0,1\}^n$ applied to a data block with size *n* defined by the key size *k*, expressed in bits.

97. In case the TOE implements cryptographic primitives based on block ciphers:

- The **TOE shall** only implement agreed block ciphers according to the table below:

Agreed Block Ciphers			
Primitive	**Parameter's Sizes**	**R/L**	**Notes**
AES [FIPS-197] [ISO-18033-3]	k = 128 bits	R	
	k = 192 bits	R	
	k = 256 bits	R	

Table 7. Agreed Block Ciphers

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/BlockCipher	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The parameterization used for the block cipher cryptographic primitives implemented by the TOE is considered as agreed upon according to Table 7. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.1.2 Stream Cipher

98. A synchronous binary stream cipher allows encrypting a plaintext of arbitrary length *L* bits by deriving a binary *L*-bit keystream sequence from a *k*-bit key and an *n*-bit initialization value, and bitwise combining modulo 2 the plaintext sequence and the keystream sequence. The obtained *L*-bit sequence represents the ciphertext.

99. In case the TOE implements cryptographic primitives based on stream ciphers:

- The **TOE shall** only implement agreed stream ciphers according to the table below and follow the specific notes:

Agreed Stream Ciphers			
Primitive	**Parameter's Sizes**	**R/L**	**Notes**
ChaCha20 [RFC-8439]	k = 256 bits nonce = 96 bits 20 rounds	R	Note 1

Table 8. Agreed Stream Cipher

Note 1 [ChaCha20 Implementation]	CL1, CL2 and CL3
The ChaCha20 stream cipher primitive shall always be implemented in conjunction with the Poly1305 message authentication code as an authenticated encryption cryptographic construction to provide confidentiality, integrity, and authenticity to the processed information.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/StreamCipher	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The parameterization used for the stream cipher cryptographic primitives implemented by the TOE is considered as agreed upon according to Table 8. - The stream cipher cryptographic primitives implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.1.3 Hash Functions

100. Cryptographic hash functions are keyless functions that take a bit string of arbitrary length as input and produce a fixed-length hash value.

101. In case the TOE implements cryptographic primitives based on hash functions:

- The **TOE shall** only implement agreed hash functions according to the table below and follow the specific notes:

Agreed Hash Functions			
Primitive	**Parameter's Sizes**	**R/L**	**Notes**
SHA-2 [FIPS-180-4] [ISO-10118-3]	h = 256 bits (SHA2-256)	R	
	h = 384 bits (SHA2-384)	R	
	h = 512 bits (SHA2-512)	R	
	h = 256 bits (SHA2-512/256)	R	
SHA-3 [FIPS-202]	h = 256 bits (SHA3-256)	R	
	h = 384 bits (SHA3-384)	R	
	h = 512 bits (SHA3-512)	R	
BLAKE2b [RFC-7693]	h = 512 bits	R	Note 2

Table 9. Agreed Hash Functions

Note: Although SHA-1 is not an agreed hash function, a particular message authentication code whose construction is based on SHA-1, namely HMAC-SHA-1, is accepted as a legacy scheme.

Note 2 [Agreed BLAKE2b]	CL1, CL2 and CL3
The BLAKE2b cryptographic primitive shall always be implemented as the underlying primitive of the Argon2id cryptographic construction for password hashing purposes. Therefore, their standalone use is considered not recommended.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/Hash	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the hash cryptographic primitives implemented by the TOE is considered as agreed upon according to Table 9. - The hash cryptographic primitives implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.1.4 Secret Sharing

102. Secret sharing (also called secret splitting) refers to methods for distributing a secret amongst a group of participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number, of possibly different types, of shares are combined; individual shares are of no use on their own.

103. In case the TOE implements cryptographic primitives based on secret sharing:

- The **TOE shall** only implement agreed secret-sharing primitives according to the table below:

Agreed Secret Sharing Primitives		
Primitive	**R/L**	**Notes**
Shamir's Secret Sharing [Sha79]	R	

Table 10. Agreed Secret Sharing Primitives

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/SecretSharing	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the secret sharing cryptographic primitives implemented by the TOE is considered as agreed upon according to Table 10. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.1.5 Extendable-Output Functions (XOF)

104. Extendable-output functions (XOFs) are an extension of the cryptographic hash functions that can generate outputs of arbitrary length, which is a significant departure from fixed-length output hash functions. They find their utility in various cryptographic applications where variable or very long hash values are desirable.

105. Of particular note are those based on Keccak, which employs a unique sponge construction that allows arbitrary output lengths, facilitating a variety of cryptographic applications. Keccak is known for its robust security against cryptographic attacks, its efficiency in hardware implementations, and its versatility, making it a valuable asset in modern cryptographic practices.

106. In case the TOE implements cryptographic primitives based on XOF:

- The **TOE shall** only implement agreed XOF primitives according to the table below and follow the specific notes:

Agreed XOF Primitives			
Primitive	**Security Strength**	**R/L**	**Notes**
SHAKE [FIPS-202]	s = 128 bits (SHAKE128)	R	Note 3
	s = 256 bits (SHAKE256)	R	Note 3
cSHAKE [SP800-185]	s = 128 bits (cSHAKE128)	R	Note 3
	s = 256 bits (cSHAKE256)	R	Note 3

Table 11. Agreed XOF Primitives

Note 3 [Agreed XOF]	CL1, CL2 and CL3
The XOF cryptographic primitives shall always be implemented as underlying primitives of cryptographic constructions such as KMAC or XMSS. Therefore, their standalone use is considered not recommended.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/XOF	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the XOF cryptographic primitives implemented by the TOE is considered as agreed upon according to Table 11. - The XOF cryptographic primitives implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.2 Symmetric Constructions

107. Symmetric constructions are built upon symmetric elementary primitives and enable them to address a large variety of security objectives. The security of keyed symmetric schemes relies on the confidentiality and integrity of a secret key shared by legitimate parties.

3.3.1.1.2.1 Symmetric Encryption (Confidentiality Only)

108. Confidentiality modes are schemes that provide an encryption procedure that transforms the plaintext into ciphertext using a secret key, and a decryption procedure that allows the plaintext to be recovered from the ciphertext and the key. They are based on a block cipher cryptographic primitive.

109. In case the TOE implements symmetric encryption cryptographic constructions to provide only confidentiality to the processed information:

- The **TOE shall** only implement agreed symmetric encryption constructions according to the table below and follow the specific notes:

Agreed Symmetric Encryption (Confidentiality Only) Constructions		
Operation Mode	**R/L**	**Notes**
CBC **[SP800-38A]** **[ISO-10116]**	R*	Note 4 Note 5 Note 7
CBC-CS **[SP800-38A-Addendum]**	R*	Note 4 Note 5
CFB **[SP800-38A]** **[ISO-10116]**	R*	Note 4 Note 5 Note 7
CTR **[SP800-38A]** **[ISO-10116]**	R*	Note 4 Note 5 Note 6
OFB **[SP800-38A]** **[ISO-10116]**	R*	Note 4 Note 5 Note 6

Table 12. Agreed Symmetric Encryption Cryptographic Constructions

Note 4 [IV Type]	**CL2 and CL3**
The encryption scheme must either be probabilistic and generate a random initialization vector to bootstrap encryption or require an additional input, whose value can only be used once with a given key (nonce). The specifications of modes of operation describe what is expected (nonce or random IV) and implementations shall follow these specifications, e.g., CBC with a constant or more generally a predictable IV does not follow the CBC specification [SP800-38A] and is not accepted. This specific note is addressed in the **CCN-PITFALL/SymEncryption.1** evaluation test.	

Note 5 [Add Integrity]	**CL1, CL2 and CL3**
Authenticated encryption schemes provide superior privacy security guarantees than encryption-only mechanisms. As a consequence, while all encryption-only modes considered above are recommended for mode construction their standalone use is considered legacy. This is the reason why the expression R^* appears in Table 12.	

Note 6 [Stream Mode]	**CL2 and CL3**
Some symmetric confidentiality schemes operate by masking the plaintext by a keystream generated from the key and IV. When using these so-called stream modes of operation, it is of utmost importance to make sure that no two generated key streams ever overlap. This can be achieved deterministically in some modes, e.g., CTR. It is ensured with overwhelming probability in other modes, e.g., OFB mode, as long as the same IV-key pair is not used (or only with negligible probability) to encrypt two messages under the same key. This specific note is addressed in **CCN-PITFALL/SymEncryption.1** and **CCN-PITFALL/CTR** evaluation tests.	

Note 7 [Padding]	**CL2 and CL3**
Some encryption modes cannot handle naturally the encryption of a last incomplete block. For such modes, a specific operation must be performed on the last block. A widespread procedure consists of padding the plaintext with some structured data to ensure its size is a multiple of the block size. The verification of the format of the padding during decryption may leak information on the decrypted value in a way that could be used to decrypt any ciphertext. More generally, any verification performed on the format of the decrypted ciphertext may leak information. Vendors/testers should ensure that the implementation of decryption does not provide an attacker with any such padding or format oracle. An alternative to the application of the padding scheme is the use of ciphertext stealing [SP800-38A-Addendum]. This specific note is addressed in the **CCN-PITFALL/SymEncryption.2** evaluation test.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/SymEncryption	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the symmetric encryption cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 12. - The symmetric encryption cryptographic constructions implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned.	**I1** **I2** **I6**

CCN-AGREED/SymEncryption	Inputs
In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	

3.3.1.1.2.2 Disk Encryption

110. Encryption modes require an expansion of the data to be ciphered of at least one block due to the addition of an initialization vector or nonce. Moreover, they cannot efficiently decrypt a specific part of the ciphertext. Because of these topics, there exist deterministic modes for disk encryption.

111. In case the TOE implements symmetric encryption cryptographic constructions to perform disk encryption operations:

- The **TOE shall** only implement agreed disk encryption constructions according to the table below and follow the specific notes:

Agreed Disk Encryption Constructions		
Operation Mode	**R/L**	**Notes**
XTS [SP800-38E]	R	Note 8 Note 9

Table 13. Agreed Disk Encryption Cryptographic Constructions

Note: The XTS mode supports key lengths of 256 and 512 bits.

Note 8 [Unique Tweak]	CL2 and CL3
The tweak value used to encrypt each complete or incomplete block position in each disk sector shall be unique, i.e., a (set of) disk(s) encrypted under the same key shall never contain two distinct blocks encrypted under the same key and the same tweak value. In the former sentence, blocks are meant as *n*-bit elements of the alphabet of the block cipher. This specific note is addressed in the **CCN-PITFALL/DiskEncryption** evaluation test.	

Note 9 [XTS Key]	CL2 and CL3
An implementation of XTS-AES that improperly generates the key so that $key_1 = key_2$ is vulnerable to a chosen ciphertext attack. Therefore, key_1 and key_2 shall be generated and/or established independently according to the rules for component symmetric keys from [SP800-133] rev2, section 6.3. The TOE shall check explicitly that $key_1 \neq key_2$ before using the keys in the XTS algorithm to process data with them. This specific note is addressed in the **CCN-PITFALL/XTS** evaluation test.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/DiskEncryption	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the disk encryption cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 13. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned and that it complies with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned and that it complies with the specific notes.	**I1** **I2** **I6**

3.3.1.1.2.3 Integrity Modes: Message Authentication Code

112. These schemes provide data integrity and data origin authentication based on symmetric mechanisms. They contain a message authentication code (MAC) generation function (inputs: a secret key K and a message m, output: μ), and a MAC verification function (inputs: K, m, μ, output: True or False).

113. In case the TOE implements MAC cryptographic constructions:

- The **TOE shall** only implement agreed MAC cryptographic constructions according to the table below and follow the specific notes:

Agreed MAC Constructions			
Scheme	**Parameter's Sizes**	**R/L**	**Notes**
CMAC **[SP800-38B]** **[ISO-9797-1]**		R	Note 10 Note 11

Agreed MAC Constructions			
Scheme	**Parameter's Sizes**	**R/L**	**Notes**
CBC-MAC [ISO-9797-1]		R	Note 10 Note 11 Note 12
HMAC [RFC-2104] [ISO-9797-2]	k ≥ 128 bits	R	Note 10 Note 11
	k ≥ 100 bits	L	Note 10 Note 11
HMAC-SHA-1 [RFC-2104] [ISO-9797-2] [FIPS-180-4]	k ≥ 100 bits	L[2030]	Note 10 Note 11 Note 13
GMAC [SP800-38D]		R	Note 14 Note 15 Note 16
KMAC-128 [SP800-185]	k ≥ 128 bits	R	Note 17
KMAC-256 [SP800-185]	k ≥ 256 bits	R	Note 17
Poly1305 [RFC-8439]		R	Note 18

Table 14. Agreed MAC Constructions

Note 10 [MAC Truncation 96 bits]	**CL1, CL2 and CL3**
Unless stated otherwise explicitly for a given mechanism, the truncation of a MAC generated by an agreed MAC mechanism to at least 96 bits is agreed. A necessary condition, this may not be sufficient for specific MAC schemes, e.g., GMAC (see Note 16).	

Note 11 [MAC Truncation 64 bits]	**CL1, CL2 and CL3**
Unless stated otherwise explicitly for a given mechanism, the truncation of a MAC generated by an agreed MAC mechanism to at least 64 bits is considered legacy under the condition that the maximal number of MAC verifications performed for a given key over its lifetime can be bound by 2^{20}.	

Note 12 [Fixed Input Length]	**CL2 and CL3**
CBC-MAC is agreed upon only in contexts where the sizes of all the inputs for which MAC is computed under the same key are identical. Trivial length extension forgeries can be performed when variable length inputs are allowed. This specific note is addressed in the **CCN-PITFALL/CBCMAC** evaluation test.	

Note 13 [HMAC-SHA-1]	**CL1, CL2 and CL3**
The HMAC construction does not require the collision resistance of the underlying hash function. For the time being, HMAC-SHA-1 is considered an acceptable legacy mechanism, even though SHA-1 is not considered an acceptable general-purpose hash function. It is recommended however to phase out HMAC-SHA-1.	

Note 14 [GMAC-GCM Nonce]	**CL2 and CL3**
The IV must be managed within the security perimeter of the authenticated encryption process. For example, it is crucial to ensure that no adversary can cause the same IV to be reused to protect different (plaintext, associated data) pairs under the same key.	

Note 15 [GMAC-GCM Options]	**CL1, CL2 and CL3**
Only the following GCM options are agreed upon: the IV length must be equal to 96 bits; the deterministic IV construction method [SP800-38D] section 8.2.1 must be used; the MAC length must be 128 bits.	

Note 16 [GMAC-GCM Bounds]	**CL1, CL2 and CL3**
As the unforgeability bounds of the agreed GMAC and GCM options of Note 15 are not optimal, the MAC length must be at least 128 bits. Thus, no truncation to a final MAC length, such as 96 bits, must be performed.	

Note 17 [KMAC Security Settings]	**CL1, CL2 and CL3**
KMAC is not allowed to be used in XOF mode.	

Note 17 [KMAC Security Settings]	CL1, CL2 and CL3
It is important to note that Note 10 and Note 11 also apply to KMAC schemes. However, they will be applied directly to the output length of KMAC, without applying truncation.	

Note 18 [Poly1305 Implementation]	CL1, CL2 and CL3
The Poly1305 message authentication code construction shall be always implemented in conjunction with the ChaCha20 stream cipher primitive as an authenticated encryption cryptographic construction to provide confidentiality, integrity, and authenticity to the processed information.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/MAC	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the MAC cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 14. - The MAC cryptographic constructions implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.2.4 Symmetric Entity Authentication Schemes

114. These schemes allow an entity to prove its identity to a correspondent by demonstrating its knowledge of a secret. They are interactive by nature and generally consist of using a MAC scheme or an encryption scheme in a random challenge-response protocol. The same key should not be used by integrity modes and symmetric entity authentication schemes.

115. In case the TOE implements entity authentication cryptographic constructions based on challenges:

- The **TOE shall** only implement agreed entity authentication challenges according to the table below and follow the specific notes:

Agreed Entity Authentication Challenges		
Challenge Size	**R/L**	**Notes**
l ≥ 125 bits	R	Note 19
96 bits ≤ l ≤ 125 bits	L	Note 19

Table 15. Agreed Entity Authentication Challenges

Note 19 [Challenge-Response Protocol]	**CL2 and CL3**
The tester must ensure that a challenge cannot be replayed with non-negligible probability. The challenge could for example be implemented by a random value, whose size is large enough, and that is generated by the tester.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/EntityAuth	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the symmetric entity authentication cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 15. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned and that it complies with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned and that it complies with the specific notes.	**I1** **I2** **I6**

3.3.1.1.2.5 Authenticated Encryption (AE)

116. The purpose of authenticated encryption (AE) is to achieve confidentiality, integrity, and data origin authentication of messages. An AE scheme provides an encryption function that transforms a plaintext into a ciphertext using a given key, and a decryption function that retrieves the plaintext from the ciphertext and the key. In comparison with an encryption-only scheme, an AE scheme decryption function may fail to return a decrypted value if the ciphertext does not respect some redundancy pattern, usually, some part of the ciphertext acts as a verification value that is checked during decryption.

117. An extra feature is often provided, namely combining the authentication of the encrypted data with the authentication of additional unencrypted data. Authenticated encryption with that feature is often referred to as Authenticated Encryption with Associated Data (AEAD). An AE or AEAD scheme may result from the combination of an encryption scheme and a message authentication code.

118. In case the TOE implements authenticated encryption cryptographic constructions:

- The **TOE shall** only implement agreed authenticated encryption constructions according to the table below and follow the specific notes:

Agreed Authenticated Encryption Constructions		
Scheme	**R/L**	**Notes**
Encrypt-then-MAC [BN00]	R	Note 20 Note 21
CCM [SP800-38C] [ISO-19772]	R	Note 20 Note 21
GCM [SP800-38D] [ISO-19772]	R	Note 14 Note 15 Note 16 Note 20 Note 21 Note 22
EAX [ISO-19772]	R	Note 20 Note 21
ChaCha20-Poly1305 [RFC-8439]	R	Note 23

Table 16. Agreed Authenticated Encryption Constructions

Note 20 [Authenticated Encryption Schemes]	**CL1, CL2 and CL3**
It should be noted that the authorized symmetric authenticated encryption schemes are all cryptographic constructions. Some of them use a single block cipher as a primitive, such as CCM or GCM; others are based on a primitive encryption scheme and on a message authentication scheme, which describes only how they are composed; for example, Encrypt-then-MAC. In such cases, the primitives must be authorized. It is important to note that Note 10 and Note 11 also apply to symmetric authenticated encryption schemes.	

Note 21 [Decryption Order]	**CL2 and CL3**
If the integrity of the ciphertexts is not properly checked before decryption, the vendor/tester should ensure that the implementation does not instantiate any padding or other error oracle. This means that the implementation does not give away any information on the padding or the format of the plaintext	

Note 21 [Decryption Order]	**CL2 and CL3**
obtained through the decryption of an arbitrary ciphertext. Otherwise, an adversary may be able to decrypt a target ciphertext by exploiting, e.g., error messages, or time side-channel information. Plaintexts should never be sent to consumer applications before their integrity has been checked. This specific note is addressed in the **CCN-PITFALL/AuthEncryption** evaluation test.	

Note 22 [GCM Plaintext Length]	**CL2 and CL3**
By specification, at each invocation of GCM, the length of the plaintext must be at most $2^{32} - 2$ blocks of the underlying block cipher. Checking that this maximal length is not exceeded is required in environments where this might potentially happen. This specific note is addressed in the **CCN-PITFALL/GCM.1** evaluation test.	

Note 23 [ChaCha20-Poly1305 IV]	**CL2 and CL3**
The IV shall be processed within the boundary of the authenticated encryption scheme. It is essential to ensure that an attacker can never cause the same IV to be used to protect two different pairs of messages and associated data with the same key. The utilization of an IV can affect confidentiality.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/AuthEncryption	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the authenticated encryption cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 16. - The authenticated encryption cryptographic constructions implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.2.6 Key Protection

119. The key protection mechanisms enable to secure storage or transmission of keys, ensuring the confidentiality, integrity, and data origin authentication of the key.

120. In case the TOE implements key protection cryptographic constructions:

- The **TOE shall** only implement agreed key protection constructions according to the table below and follow the specific notes:

Agreed Key Protection Constructions		
Scheme	**R/L**	**Notes**
SIV [RFC-5297]	R	
AES-Keywrap [SP800-38F] Mechanisms KW and KWP	R	
Authenticated Encryption Construction (Table 16)	R	*Notes according to the Table 16*

Table 17. Agreed Key Protection Constructions

Note: The SIV mode supports key lengths of 256, 384, and 512 bits.

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/KeyProtection	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the key protection cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 17 and/or Table 16. - The authenticated encryption cryptographic constructions implemented by the TOE (used for key protection capabilities) comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.2.7 Key Derivation Functions (KDF)

121. A key derivation mechanism enables to derivation of several keys from a single master key. It generally takes three arguments as input, a secret value *K*, a

(possibly) public value *N*, and a length *n*, and then generates *n* bits that can be split into several keys that appear to be independent.

122. In case the TOE implements key derivation cryptographic constructions:

- The **TOE shall** only implement agreed key derivation functions according to the table below and follow the specific notes:

Agreed Key Derivation Constructions		
Scheme	**R/L**	**Notes**
NIST SP800-56 A/B/C [SP800-56A] [SP800-56B] [SP800-56C]	R	
NIST SP800-108 [SP800-108]	R	
ANSI-X9.63 KDF [ANSI-X9.63]	R	
PBKDF2 [RFC-8018] [SP800-132]	R	Note 24
HKDF [RFC-5869]	R	

Table 18. Agreed Key Derivation Constructions

Note 24 [PBKDF2 PRF]	**CL2 and CL3**
PBKDF2 is built over a pseudo-random function, that can be instantiated using a MAC generation function. This pseudo-random function shall be an agreed mechanism. In the case where HMAC is used, care has to be taken regarding the key length. Indeed, if the HMAC key is longer than the hash function message block length, the key is hashed. This pre-hashing in HMAC can lower the effective entropy of the key derived using PBKDF2.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/KeyDerivation	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the key derivation cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 18. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned and that it complies with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned and that it complies with the specific notes.	**I1** **I2** **I6**

3.3.1.1.2.8 Password Protection/Hashing Mechanisms

123. Password protection/hashing mechanisms associate a password with a verification value. Systems relying on passwords to authenticate users store these verification values. This enables them to test passwords without storing them. Even in the case of the verification values being compromised, an adversary should not be able to recover a strong password.

124. In case the TOE implements password protection/hashing cryptographic constructions:

- The **TOE shall** only implement agreed password protection/hashing functions according to the table below and follow the specific notes:

Agreed Password Protection/Hashing Constructions		
Scheme	**R/L**	**Notes**
Argon2id [RFC-9106]	R	Note 25 Note 26
PBKDF2 [RFC-8247]	R	Note 27 Note 28
SCRYPT [RFC-7914]	R	

Table 19. Agreed Password Protection/Hashing Constructions

Note 25 [Use of Argon2id]	**CL1, CL2 and CL3**
There are two recommended configurations for the use of Argon2id depending on the environment. The first configuration, default in all environments, is the variant with parameters number of passes t = 1 and 2 GB of memory because it is secure against side-channel attacks and maximizes the cost of confrontation on dedicated brute-force hardware. The second configuration, default for memory-limited environments, is the variant with parameters number of passes t = 3 and 64 MB of memory.	

Note 26 [Argon2id Parameters]
In addition to following one of the recommended configurations described in Note 25, the following parameterization shall be complied with: - Password length not greater than $2^{32} - 1$ bytes - Degree of parallelism (p) of 4 lanes - 128-bit Salt and unique for each password - 256-bit Tag If optional parameters such as secret value (K) or associated data (X) are used, the length must not be greater than $2^{32} - 1$ bytes.

Note 27 [Number of Iterations]	**CL1, CL2 and CL3**
Password hashing mechanisms offer parameters controlling the complexity of the hashing execution. This allows to increase in the work factor of brute-force attacks: a legitimate use of the password hashing requires only one execution, while a brute-force attack requires a large number of executions. Thus, the number of iterations of PBKDF2 should be selected as large as possible so that it does not impede legitimate use.	

Note 28 [Salt]	**CL1, CL2 and CL3**
Salt is a random value, generated when the password is registered, and that is stored along with the password verification value. Salting a password hashing mechanism counters pre-computation attacks. The length of the salt shall be at least 128 bits.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/PasswordMechanisms	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the password protection/hashing cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 19. - The password protection/hashing cryptographic constructions implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.3 Asymmetric Elementary Primitives

125. In asymmetric cryptography, security is based on the discrepancy between the computational difficulties of two mathematical problems: one is easy whereas the other is hard. That asymmetry is needed in order to ensure the possibility of generating pairs of private and public keys for which it is computationally hard to recover the private key from the public key. There should be no polynomial mechanism to deduce the private key from the public key.

126. This section contains the asymmetric atomic primitives and corresponding mathematical problems, moreover, the next section describes the asymmetric constructions that can be achieved by building upon these primitives.

3.3.1.1.3.1 RSA/Integer Factorization

127. The RSA primitive consists of a public permutation parametrized by a public key and a private inverse permutation parametrized by the associated private key. This primitive is invoked in various RSA encryption or RSA signature schemes addressed in the next section. These schemes also specify padding and redundancy check conventions.

128. Note that the primitive alone must not be considered in a complete encryption or signature scheme since its use without extra conventions would be highly insecure.

129. In case the TOE implements asymmetric cryptographic primitives based on integer factorization:

- The **TOE shall** only implement agreed integer factorization schemes according to the table below:

Agreed Asymmetric Primitives based on Integer Factorization			
Primitive	**Parameter's Sizes**	**R/L**	**Notes**
RSA [RSA78]	n ≥ 3000 $\log_2 e > 16$	R	
	n ≥ 1900 $\log_2 e > 16$	L[2025]	

Table 20. Agreed Asymmetric Primitives based on Integer Factorization

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/RSA	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The parameterization used for the asymmetric cryptographic primitives based on integer factorization implemented by the TOE is considered as agreed upon according to Table 20. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

130. In case the TOE implements RSA key pair generation:

- The **TOE shall** only implement agreed RSA key pair generation schemes according to the table below and follow the specific notes. This also implies the implementation of:
  - An agreed prime random number generation method for the generation of primes p and q (see section 3.3.1.1.5.3.2 "Random Prime Number Generation"), that makes use of the Miller-Rabin as the primality test.
  - An agreed random uniform distribution method for the generation of the testing candidates and the basis parameter in each iteration of the Miller-Rabin algorithm (see section 3.3.1.1.5.3.1 "Random Uniform Modular Distributions").

Agreed RSA Key Generation Schemes	
Scheme	**Notes**
RSA Key-Pair Generation **[SOGIS-HEP] Append. B.3** **[CCN-STIC-221] Algorithm 6**	Note 29

Table 21. Agreed RSA Key Generation Schemes

Note 29 [Primes]	**CL2 and CL3**
As described in the scheme, the generation of primes *p* and *q* shall be done by means of an agreed prime number generation method (see section 3.3.1.1.5.3.2 "Random Prime Number Generation"). The conditions for *Test(p)* and *Test(q)* to be employed by the prime generation method used to generate *p* and *q* are defined by the agreed scheme.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/RSAKeys	**Inputs**
In the case of **CL2** evaluations, use **I1** and operate the TOE (using **I2**) to verify that: For the key pair generation: - The scheme used for the RSA key pair generation processes implemented by the TOE is considered as agreed upon according to Table 21. - The RSA key pair generation schemes implemented by the TOE comply with the specific notes. For the generation of primes *p* and *q*: - The method used for the random prime generation implemented by the TOE is considered as agreed upon according to Table 32. - The random prime generation methods implemented by the TOE comply with the specific notes. For the generation of the testing candidates and the Miller-Rabin basis parameter in each iteration: - The scheme used for the random uniform modular distribution implemented by the TOE is considered as agreed upon according to Table 31. - The random uniform modular distribution methods implemented by the TOE comply with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.3.2 Multiplicative Discrete Logarithm Problem

131. The security of several asymmetric cryptographic schemes relies on the difficulty of the discrete logarithm problem (DLP) in the multiplicative group of finite fields, in comparison to the easiness of modular exponentiation (MODP) in finite fields.

132. In case the TOE implements asymmetric cryptographic primitives based on the multiplicative discrete logarithm problem over a finite prime field (FFDLOG):

- The **TOE shall** only implement agreed FFDLOG schemes according to the table below:

Agreed Asymmetric Primitives based on FFDLOG			
Primitive	**Parameter's Sizes**	**R/L**	**Notes**
MODP [RFC-3526]	3072 bits group	R	
	4096 bits group	R	
	6144 bits group	R	
	8192 bits group	R	
	2048 bits group	L[2025]	
FFDHE [RFC-7919]	3072 bits group	R	
	4096 bits group	R	
	6144 bits group	R	
	8192 bits group	R	
	2048 bits group	L[2025]	

Table 22. Agreed Asymmetric Primitives based on FFDLOG

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/FFDLOG	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The parameterization used for the asymmetric cryptographic primitives based on FFDLOG implemented by the TOE is considered as agreed upon according to Table 22. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

133. In case the TOE implements FFDLOG key pair generation:

- The **TOE shall** only generate private keys using an agreed random uniform modular distribution according to Table 31 and follow the associated specific notes.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/FFDLOGKeys	Inputs
In the case of **CL2** evaluations, use **I1** and operate the TOE (using **I2**) to verify that: - The scheme used for the FFDLOG private key generation processes implemented by the TOE uses an agreed random uniform modular distribution according to Table 31. - The FFDLOG private key generation schemes implemented by the TOE comply with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.3.3 Additive Discrete Logarithm Problem

134. The difficulty of the discrete logarithm problem can also be considered in the group of rational points of an elliptic curve over a finite field. In this case, the problem is called the discrete logarithm problem on elliptic curves, elliptic, additive, or ECDLP (Elliptic Curve Discrete Logarithm Problem), as opposed to the previous case, in which the operation considered was multiplication in a finite group. In this case, the primitive is known as the discrete logarithm on elliptic curves and in abbreviated form as ECDLOG (Elliptic Curve Discrete Logarithm).

135. In comparison with the discrete logarithm problem in finite fields, there are a couple of choices to be made in the case of elliptic curves: first, the finite field over which the elliptic curve will be defined, and second the elliptic curve itself. As in the case of finite fields, only elliptic curves defined over prime fields are agreed upon.

136. In case the TOE implements asymmetric cryptographic primitives based on the additive discrete logarithm problem over elliptic curves (ECDLOG):

- The **TOE shall** only implement agreed ECDLOG schemes according to the table below and follow the specific notes:

Agreed Asymmetric Primitives based on ECDLOG			
Curve Family	**Curve**	**R/L**	**Notes**
Brainpool [RFC-5639]	BrainpoolP256r1	R	Note 30 Note 31 Note 32
	BrainpoolP384r1	R	
	BrainpoolP512r1	R	
NIST [SP800-186]	NIST P-256	R	Note 30 Note 31 Note 32 Note 33
	NIST P-384	R	
	NIST P-521	R	
FR [ANS11]	FRP256v1	R	Note 30 Note 31 Note 32
Montgomery [SP800-186]	Curve25519	R	Note 30 Note 31 Note 32
	Curve448	R	
Twisted Edwards [SP800-186] [FIPS-186-5]	Edwards25519	R	Note 30 Note 31 Note 32
	Edwards448	R	

Table 23. Agreed Asymmetric Primitives based on ECDLOG

Note 30 [Points on the Curve]	**CL2 and CL3**
Special precautions should be taken to ensure that manipulated points lie on the curve, i.e., they verify the curve equation. This specific note is addressed in the **CCN-PITFALL/ECDLOG** evaluation test.	

Note 31 [Points on a Subgroup]	**CL2 and CL3**
In case a curve with non-prime order is used, it should be ensured that the manipulated points lie in the intended subgroup and have order divisible by r. This specific note is addressed in the **CCN-PITFALL/ECDLOG** evaluation test.	

Note 32 [Prime Order]	**CL2 and CL3**
If the subgroup order is chosen to be prime, i.e., $q = r$, and such that r^2 does not divide $\#E(GF(p))$, the verifications for Note 31 boil down to checking that the manipulated points are of exact order r.	

Note 33 [Prime Selection]	CL2 and CL3
The special form of the prime number *p* used to construct the finite field *GF(p)* makes side-channel attacks more efficient than with a random prime (and not only because the arithmetic of the underlying finite field is faster).	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/ECDLOG	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The parameterization used for the asymmetric cryptographic primitives based on ECDLOG implemented by the TOE is considered as agreed upon according to Table 23. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned and that it complies with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned and that it complies with the specific notes.	**I1** **I2** **I6**

137. In case the TOE implements ECDLOG key pair generation:

- The **TOE shall** only generate private keys using an agreed random uniform modular distribution according to Table 31 and follow the associated specific notes.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/ECDLOGKeys	Inputs
In the case of **CL2** evaluations, use **I1** and operate the TOE (using **I2**) to verify that: - The scheme used for the ECDLOG private key generation processes implemented by the TOE uses an agreed random uniform modular distribution according to Table 31. - The ECDLOG private key generation schemes implemented by the TOE comply with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.4 Asymmetric Constructions

138. Asymmetric mathematical problems can be used to build asymmetric schemes. Typically, in such schemes, each user has attributed a key pair, consisting of a public key *pk* that can be published, and an associated private key *sk* that should remain confidential. The difficulty of the underlying mathematical problem guarantees that neither the private key can be recovered from the public key, nor the sensitive operation of the scheme (e.g., decryption, or signature generation) can be performed without the private key.

139. The security of asymmetric schemes relies on the security of a mathematical problem and can also rely on the security of a symmetric primitive or scheme. For a cryptographic construction to be agreed upon, it has to be based on agreed primitives. In particular, the requirements on parameter sizes established in the previous section also apply to the schemes in this section.

140. The security of keyed asymmetric schemes relies on the confidentiality and integrity of the private key and the integrity and data origin authentication of the public key.

3.3.1.1.4.1 Asymmetric Encryption Schemes

141. An asymmetric encryption scheme contains two functions. The encryption transforms any message *m*, using the public key *pk*, into a ciphertext *c*. The decryption function enables to recover *m* from *c* and *sk*.

142. In case the TOE implements asymmetric encryption cryptographic constructions:

- The **TOE shall** only implement agreed asymmetric encryption schemes according to the table below and follow the specific notes:

Agreed Asymmetric Encryption Constructions			
Primitive	**Scheme**	**R/L**	**Notes**
RSA	**OAEP (PKCS#1v2.1) [RFC-8017]**	R	Note 34 Note 35
	PKCS#1v1.5 [RFC-8017]	L	Note 34

Table 24. Agreed Asymmetric Encryption Constructions

Note 34 [Random Padding]	**CL2 and CL3**
The asymmetric encryption schemes use randomized padding that shall be generated by an agreed deterministic random number generator.	

Note 35 [Padding-OAEP Attack]	CL2 and CL3
In case the OAEP decryption procedure is not correctly implemented, that is to say, the checks performed by EME-OAEP decoding are not performed in the specified order, RSA OAEP may also be vulnerable to an Oracle attack. This specific note is addressed in the **CCN-PITFALL/AsymEncryption** evaluation test.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/AsymEncryption	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the asymmetric encryption cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 24. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned and that it complies with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned and that it complies with the specific notes.	**I1** **I2** **I6**

3.3.1.1.4.2 Digital Signature

143. A digital signature scheme offers a signature generation function (inputs: a private key *sk* and a message *m*, output: a signature σ), and a verification function (inputs: the public key *pk*, the message *m*, and the signature σ, output: True or False). Digital signature schemes offer data authentication and non-repudiation.

144. In case the TOE implements digital signature cryptographic constructions:

- The **TOE shall** only implement agreed digital signature schemes according to the table below and follow the specific notes:

Agreed Digital Signature Constructions			
Primitive	**Scheme**	**R/L**	**Notes**
RSA	**PSS (PKCS#1v2.1) [RFC-8017] [ISO-9796-2]**	R	Note 36
FFDLOG	**KCDSA [ISO-14888-3]**	R	Note 36 Note 38

Agreed Digital Signature Constructions			
Primitive	**Scheme**	**R/L**	**Notes**
	Schnorr [ISO-14888-3]	R	Note 36 Note 38
	DSA [FIPS-186-4] [ISO-14888-3]	R	Note 36 Note 38
ECDLOG	**EC-KCDSA [ISO-14888-3]**	R	Note 36 Note 38
	EC-DSA [FIPS-186-5] [ISO-14888-3]	R	Note 36 Note 38
	EC-GDSA [TR-03111]	R	Note 36 Note 38
	EC-Schnorr [ISO-14888-3]	R	Note 36 Note 38
	EdDSA [FIPS-186-5] [RFC-8032]	R	
RSA	**PKCS#1v1.5 [RFC8017] [ISO-9796-2]**	L	Note 36 Note 37
Hash and XOF	**XMSS [RFC-8391] [SP800-208]**	R	Note 39 Note 40 Note 41 Note 42 Note 43 Note 44
Lattice	**ML-DSA [FIPS-204]**	R	Note 45
Lattice	**Falcon [PFH+20]**	R	Note 45
Hash and XOF	**SLH-DSA [FIPS-205]**	R	Note 45

Table 25. Agreed Digital Signature Constructions

Note 36 [Hash Function]	**CL1, CL2 and CL3**
The schema is agreed as long as the underlying hash function is agreed.	

Note 37 [PKCS Format Check]	**CL2 and CL3**
Format checks should be carefully implemented to avoid attacks *à la Bleichenbacher*.	

Note 38 [DSA Randomness]	**CL2 and CL3**
In DSA and its elliptic curve variants, the digital signature generation procedure generates a random value k. This random value shall be chosen according to a uniform distribution making use of the agreed schemes (see section 3.3.1.1.5.3.1 “Random Uniform Modular Distributions”). In the case of private keys generation, the cryptographic evaluation tests **CCN-AGREED/FFDLOGKeys** or **CCN-AGREED/ECDLOGKeys** shall be considered.	

Note 39 [XMSS Underlying Primitives]	**CL1, CL2 and CL3**
This scheme shall only be used with SHA2-256 and SHAKE256/256 underlying primitives.	

Note 40 [Stateful Post-quantum Digital Signatures$_1$]	**CL1, CL2 and CL3**
This scheme shall only be used for FW/SW verification.	

Note 41 [Stateful Post-quantum Digital Signatures$_2$]	**CL2 and CL3**
In XMSS the private key is updated each time a message is signed. If the key is stored in non-volatile memory, it shall be ensured that the private key is updated in non-volatile memory before exporting the corresponding signature. This specific note is addressed in the **CCN-PITFALL/XMSS.1** evaluation test.	

Note 42 [Precautions for XMSS Backups]	**CL2 and CL3**
The security of XMSS signatures is weakened when the same private key and counter are used to sign different documents. Appropriate precautions shall be taken to prevent this problem from occurring during the backup recovery process. This specific note is addressed in the **CCN-PITFALL/XMSS.2** evaluation test.	

Note 43 [Stateful Post-quantum Digital Signatures$_3$]	**CL2 and CL3**
Before generating the signature of a message, it shall be verified that the private key counter does not exceed the maximum allowed by the established parameterization. This specific note is addressed in the **CCN-PITFALL/XMSS.3** evaluation test.	

Note 44 [Stateful Post-quantum Digital Signatures$_4$]	**CL2 and CL3**
It shall be verified that the derivation of the *WOTS+* keys is performed according to [SP800-208], i.e. using the *PRF_keygen* function to avoid multi-target attacks. This specific note is addressed in the **CCN-PITFALL/XMSS.4** evaluation test.	

Note 45 [Post-quantum Digital Signatures Implementation]	**CL1, CL2 and CL3**
The ML-DSA and SLH-DSA cryptographic mechanisms shall be implemented according to the guidelines provided in the final NIST standard. The Falcon mechanism is not recommended to be implemented until NIST publishes the final standard, since until then, the mechanism may undergo variations.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/DigitalSignature	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the digital signature cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 25. - The digital signature cryptographic constructions implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.4.3 Key Establishment

145. Asymmetric key establishment schemes allow two or more parties to generate a common secret without using any pre-shared secret values. They are usually combined with asymmetric or symmetric authentication based on a public-private key pair or a shared secret key.

146. The most widely used two-party key establishment scheme has been proposed by Diffie and Hellman and relies on the discrete logarithm problem (instantiated in any suitable group).

147. In case the TOE implements key establishment cryptographic constructions:

- The **TOE shall** only implement agreed key establishment schemes according to the table below and follow the specific notes:

Agreed Key Establishment Constructions			
Primitive	**Scheme**	**R/L**	**Notes**
FFDLOG	**DH [SP800-56A] [ISO-11770-3]**	R	Note 46 Note 47 Note 50
	DLIES-KEM [ISO-18033-2]	R	Note 46 Note 47 Note 48 Note 50
ECDLOG	**EC-DH [SP800-56A] [ISO-11770-3]**	R	Note 46 Note 47 Note 50

Agreed Key Establishment Constructions			
Primitive	**Scheme**	**R/L**	**Notes**
	ECIES-KEM [ISO-18033-2]	R	Note 46 Note 47 Note 49 Note 50
Lattice	**ML-KEM [FIPS-203]**	R	Note 46 Note 51
	FrodoKEM [ISO-18033-2]	R	Note 46 Note 51

Table 26. Agreed Key Establishment Constructions

Note 46 [Authentication]	**CL2 and CL3**
These key establishment schemes are unauthenticated and may fall to man-in-the-middle attacks. In order to ensure security, it is necessary to authenticate both parties and the data exchanged during the key establishment scheme, such as public points and identities. This authentication requires long-term secrets.	

Note 47 [DH/EC-DH Subgroup Attacks]	**CL2 and CL3**
Whether the Diffie-Hellman protocol is defined over the multiplicative group of a finite field (DH) or the group of rational points of an elliptic curve (EC-DH), it should be ensured that the manipulated values lie in the intended subgroup and have a large enough order (see Note 30 and Note 31). This specific note is addressed in **CCN-PITFALL/FFKeyEstablishment** and **CCN-PITFALL/ECKeyEstablishment** evaluation tests.	

Note 48 [DLIES Key Length]	**CL1, CL2 and CL3**
A requirement to ensure the security of the DLIES is that it is impossible to determine the discrete logarithm in the subgroup generated by *g*. According to the current state of the art, the difficulty of DLP on F_P^* can be significantly reduced by precomputations that only rely on *p* and not, for example, on the chosen subgroup or its generator. Therefore, this requires that the length of the prime number *p* to be at least 3000 bits and the length of the prime number *q* to be at least 250 bits.	

Note 49 [ECIES Key Length]	**CL1, CL2 and CL3**
Given that for ECIES the same requirement must be verified as that indicated in Note 48 concerning the difficulty of solving the DLP in the subgroup generated by the point of the curve considered with a given order. The length of this order must be at least 250 bits.	

Note 50 [Uniform Distributions]	**CL2 and CL3**
DH, EC-DH, DLIES-KEM, and ECIES-KEM are probabilistic mechanisms since random values belonging to certain integer intervals are selected, for private key generation. Thus, such values must be chosen following a uniform distribution (see section 3.3.1.1.5.3.1 "Random Uniform Modular Distributions"). The cryptographic evaluation tests **CCN-AGREED/FFDLOGKeys** or **CCN-AGREED/ECDLOGKeys** shall be considered.	

Note 51 [Post-quantum KEMs Implementation]	**CL1, CL2 and CL3**
The ML-KEM cryptographic mechanism shall be implemented according to the guidelines provided in the final NIST standard. The FrodoKEM cryptographic mechanism shall be implemented according to the guidelines provided in the final ISO standard.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/KeyEstablishment	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The scheme used for the key establishment cryptographic constructions implemented by the TOE is considered as agreed upon according to Table 26. - The key establishment cryptographic constructions implemented by the TOE comply with the specific notes. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned.	**I1** **I2** **I6**

3.3.1.1.5 Random Number Generators

148. A random number generator (RNG) is composed of a true random number generator, which generates unpredictable digital data, and a deterministic component that processes this data to produce the RNG's output sequence (internal random numbers).

149. The standards AIS 20 and AIS 31 contain guidelines and requirements developed by the German Federal Office for Information Security (BSI) for the testing and evaluation of DRNGs and TRNGs. These guidelines provide a comprehensive framework for assessing the quality, security, and randomness of the output produced by RNGs.

150. Both standards are based on the document "*A Proposal for Functionality Classes for Random Number Generators*", referred to in this methodology as [FCRNG]. This defines various functionality classes, organizing and specifying the requisites that both, Deterministic and True Random Number Generators must adhere to.

151. As will be explained in the following sections, CCN requires that the RNGs employed by the TOE comply with the requirements specified in the mathematical-technical reference document [FCRNG].

Note: The mathematical-technical document [FCRNG] is currently being updated by the BSI, this guide refers to versions of the document as of June 2023 onward.

3.3.1.1.5.1 True Random Number Generators

152. A TRNG is a device or mechanism for which the output random values depend on an entropy source; a source of unpredictable data. A distinction is made between physical true random number generators (PTRNG) and non-physical true random number generators (NPTRNG). In most cases, it is necessary to employ deterministic post-processing of the digitized digital noise data as acquired from the source (raw noise data), aiming to eliminate any bias or dependence on the random output distribution.

153. A TRNG can be understood as a probabilistic procedure that provides random bits. The main objective when evaluating a TRNG is to obtain an estimate of the entropy per bit that the TRNG is capable of producing, which is a measure of the unpredictability of the random distribution on which the TRNG is based. An ideal TRNG would produce 1 bit of Shannon entropy per bit.

154. In case the TOE makes use of random number generation:

- The **TOE shall** only make use of True Random Number Generators for which the average entropy per internal random bit, h_{rate}, fulfills the following conditions:
    - In the case of PTRNGs, the average Shannon entropy per internal random bit is greater or equal to 0.9998 or the min-entropy per internal random bit is greater or equal to 0.98.

- In the case of NPTRNGs, the min-entropy per internal random bit is greater or equal to 0.98.

Note 52 [TRNG Use]	**CL2 and CL3**
Due to the difficulty of evaluating the quality of a true random source, its use should be limited to obtaining the seed and optionally reseeding a deterministic random number generator.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/TRNGEntropy	**Inputs**
In the case of **CL2** and **CL3** evaluations, verify using **I8** that: - The average entropy per internal random bit, h_{rate}, fulfills the following conditions: ▪ In the case of PTRNGs, the average Shannon entropy per internal random bit is greater or equal to 0.9998 or the min-entropy per internal random bit is greater or equal to 0.98. ▪ In the case of NPTRNGs, the min-entropy per internal random bit is greater or equal to 0.98. **Note:** In order to verify this test, the tester shall perform the **CCN-AGREED/PTRNG** or the **CCN-AGREED/NPTRNG** evaluation tests.	**I8**

Note: The internal random bits refer to the final stage of the random bits of an RNG that are ready to be output. For TRNGs, this means random bits after post-processing.

3.3.1.1.5.1.1 Physical True Random Number Generators

155. A physical true random number generator (PTRNG) is based on an entropy source that exploits physical phenomena (thermal noise, shot noise, jitter, metastability, radioactive decay, etc.) from dedicated hardware designs (using diodes, ring oscillators, etc.) or physical experiments to produce digitized random data. In many cases, it is necessary to use a deterministic post-processing of the digitally captured noise data obtained from the source (raw noise data) in order to eliminate any bias or dependency.

156. In case the TOE implements a physical true random number generator as TRNG:

- The **TOE shall** only implement PTRNGs that belong to an agreed AIS-31 functionality class according to the table below and follow the specific notes:

Agreed Physical True Random Number Generators Classes		
Functionality Class	**R/L**	**Notes**
PTG.3 [FCRNG]	R	Note 52 Note 53
PTG.2 [FCRNG]	R	Note 52 Note 53

Table 27. Agreed Physical True Random Number Generators Classes

Note 53 [PTRNG Evaluation]	**CL2 and CL3**
The PTRNG functionality classes defined in the [FCRNG] technical reference (PTG.2 and PTG.3) are nothing more than a set of requirements; if a PTRNG meets all of them, then it belongs to that functionality class. In order to prove that a PTRNG belongs to a certain functionality class, the TOE shall meet the requirements for functionality class PTG.2 or PTG.3.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/PTRNG	**Inputs**
In the case of **CL2** and **CL3** evaluations, verify in **I1** that: - The physical true random number generators implemented by the TOE belong to one of the agreed AIS-31 PTRNG functionality classes according to Table 27. - The physical true random number generators implemented by the TOE comply with the specific notes. In addition, the tester shall perform a full evaluation against the AIS-31 requirements defined in the [FCRNG] technical reference for all the PTRNGs implemented by the TOE. To do so, the tester shall: 1) Verify that, for each PTRNG defined in **I1**, the vendor has provided in **I8** all the required evidence. 2) Verify that all the data used by the vendor to perform statistical tests have been outputted by the PTRNG implemented by the TOE. This can be done by performing a witnessing session.	**I1** **I8**

CCN-AGREED/PTRNG	Inputs
3) Repeat all statistical tests performed by the vendor and check that the results are valid. 4) Verify that the Stochastic Model provided in **I8** is mathematically sound, that the model starts from the analog noise source, that the digitalization is properly assessed, and that the derived entropy bound is derived from the model. For this purpose, the tester shall follow all the derivation steps and mathematical arguments given by the vendor and check their validity. 5) Verify that the Stochastic Model conforms to the real PTRNG behavior. For this purpose, the tester shall reproduce the verification method employed by the vendor and consider it sufficient. 6) Verify that the Online Test is tailored to the Stochastic Model. 7) Verify all the AIS-31 requirements for PTG.2 or PTG.3 using **I8**. For this purpose, the tester may ask the vendor for more pieces of evidence, if needed. **Note:** The successful verification of this test implies the successful verification of the **CCN-AGREED/TRNGEntropy** evaluation test. The inverse is not true.	

3.3.1.1.5.1.2 Non-Physical True Random Number Generators

157. Just like in PTRNGs, NPTRNGs also generate truly random numbers, so they must produce sufficient entropy, but they do not use dedicated hardware. Instead, they utilize specific system resources (such as system time, RAM content, etc.) or interactions with the user (like mouse movement, keyboard input timing, etc.).

158. In case the TOE implements a non-physical true random number generator as TRNG:

- The **TOE shall** only implement NPTRNGs that belong to an agreed AIS-31 functionality class according to the table below and follow the specific notes:

Agreed Non-Physical True Random Number Generators Classes		
Functionality Class	**R/L**	**Notes**
NTG.1 [FCRNG]	R	Note 52 Note 54

Table 28. Agreed Non-Physical True Random Number Generators Classes

Note 54 [NPTRNG Evaluation]	**CL2 and CL3**
The NPTRNG functionality classes defined in the [FCRNG] technical reference (NTG.1) are nothing more than a set of requirements; if an NPTRNG meets all of them, then it belongs to NTG.1. In order to prove that an NPTRNG belongs to NTG.1, the TOE shall meet the requirements for functionality class NTG.1.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/NPTRNG	**Inputs**
In the case of **CL2** and **CL3** evaluations, verify in **I1** that: - The non-physical true random number generators implemented by the TOE belong to one of the agreed AIS-31 NPTRNG functionality classes according to Table 28. - The non-physical true random number generators implemented by the TOE comply with the specific notes. In addition, the tester shall perform a full evaluation against the AIS-31 requirements defined in the [FCRNG] technical reference for all the NPTRNGs implemented by the TOE. To do so, the tester shall: 1) Verify that, for each NPTRNG defined in **I1**, the vendor has provided in **I8** all the required evidence. 2) Verify that all the data used by the vendor to perform statistical tests have been outputted by the NPTRNG implemented by the TOE. This can be done by performing a witnessing session. 3) Repeat all statistical tests performed by the vendor and check that the results are valid. 4) Verify that the theoretical description and justification for entropy estimates provided in **I8** are mathematically sound. For this purpose, the tester shall follow all the derivation steps and mathematical and heuristic arguments given by the vendor and check their validity. 5) Verify all the AIS-31 requirements for NTG.1 using **I8**. For this purpose, the tester may ask the vendor for more pieces of evidence, if needed.	**I1** **I8**

CCN-AGREED/NPTRNG	Inputs
Note: The successful verification of this test implies the successful verification of the **CCN-AGREED/TRNGEntropy** cryptographic test. The inverse is not true.	

3.3.1.1.5.2 Deterministic Random Number Generators

159. The output of a TRNG shall be processed by a Deterministic Random Number Generator (DRNG). This is a (deterministic) cryptographic construction, built around an internal state, that can be seeded and refreshed by the output of random sources, and from which (pseudo-)random bits can be extracted.

160. In the case of functionality classes PTG.3 and NTG.1, their post-processing functions are required to be DRG.3 compliant DRNGs. So they can be seen as "hybrid RNGs" that already contain a DRNG.

161. Similarly to the case of TRNGs, the BSI's AIS-20 standard defines requirements for deterministic random number generators (DRNGs). The technical reference [FCRNG] organizes DRNG requirements into various functionality classes.

162. In case the TOE implements random number generation:

- The **TOE shall** only implement DRNGs that belong to an agreed AIS-20 functionality class according to the table below and follow the specific notes:

Agreed Deterministic Random Number Generator Classes		
Functionality Class	**R/L**	**Notes**
DRG.3 [FCRNG]	R	Note 55
DRG.4 [FCRNG]	R	Note 55

Table 29. Agreed Deterministic Random Number Generator Classes

Note 55 [DRNG Evaluation]	CL2 and CL3
The DRNG functionality classes defined in the [FCRNG] technical reference (DRG.3 and DRG.4) are nothing more than a set of requirements; if a DRNG meets all of them, then it belongs to that functionality class. In order to prove that a DRNG belongs to a certain functionality class, the TOE shall meet the requirements for functionality class DRG.3 or DRG.4.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/DRNG	Inputs
In the case of **CL2** and **CL3** evaluations, verify in **I1** that: - The deterministic random number generators implemented by the TOE belong to one of the agreed AIS-20 DRNG functionality classes according to Table 29. - The deterministic random number generators implemented by the TOE comply with the specific notes. In addition, the tester shall perform a full evaluation against the AIS-20 requirements defined in the [FCRNG] technical reference for all the DRNGs implemented by the TOE. To do so, the tester shall verify all the AIS-20 requirements for DRG.3 or DRG.4 using **I8**. **Note:** If the DRNG scheme is HMAC-DRBG or Hash-DRBG and complies with the specific notes (Note 56 or Note 57 respectively), it can be considered as a DRG.3 deterministic random number generator. Therefore, no additional verification will be required to accomplish this test.	**I1** **I8**

163. On the other hand, some of the most employed DRNG schemes are Hash-DRBG, HMAC-DRBG, and CTR-DRBG from [SP800-90A]. For Hash-DRBG and HMAC-DRBG, the document [FCRNG] provides conformity proofs to the functionality class DRG.3 under certain conditions. Such conformity proof does not exist for CTR-DRBG, which means its evaluation process would be more time-consuming.

164. It shall be noted that HMAC-DRBG and Hash-DRBG can be proven to be AIS-20 compliant by the BSI. Thus, it is highly recommended to use them in **CL2** and **CL3** levels, as it is not known whether CTR-DRBG can be regarded as AIS-20 compliant.

165. In case the TOE implements an [SP800-90A] compliant DRNG:

- The **TOE shall** only implement agreed deterministic random number generator schemes according to the table below and follow the specific notes:

Agreed Deterministic Random Number Generator Schemes		
Schemes	**R/L**	**Notes**
HMAC-DRBG [SP800-90A] [ISO-18031]	R	Note 56 Note 58 Note 59
Hash-DRBG [SP800-90A] [ISO-18031]	R	Note 57 Note 58 Note 59

Agreed Deterministic Random Number Generator Schemes		
Schemes	**R/L**	**Notes**
CTR-DRBG [SP800-90A] [ISO-18031]	R	Note 58 Note 59

Table 30. Agreed Deterministic Random Number Generators

Note 56 [DRNG HMAC-DRBG Conformity Proof]	**CL2 and CL3**

According to the conformity proof described in the technical reference [FCRNG], if the DRNG implemented by the TOE is the HMAC-DRBG as defined in the [SP800-90A] and fulfills all of the following conditions:

- The hash primitive used is one of the following: SHA2-256, SHA2-512/256, SHA2-384, SHA2-512, SHA3-256, SHA3-384 or SHA3-512.
- The *entropy_input* (seed) (as defined in [SP800-90A]) string consists of ≥ 256 bits that have been generated by a TRNG that is compliant with class PTG.2, PTG.3 or NTG.1.

Then, HMAC-DRBG can be claimed to belong to the DRG.3 functionality class.

Note 57 [DRNG Hash-DRBG Conformity Proof]	**CL2 and CL3**

According to the conformity proof described in the technical reference [FCRNG], if the DRNG implemented by the TOE is the Hash-DRBG as defined in the [SP800-90A] and fulfills all of the following conditions:

- The hash primitive used is one of the following: SHA2-256, SHA2-512/256, SHA2-384, SHA2-512, SHA3-256, SHA3-384 or SHA3-512.
- The *entropy_input* (seed) (as defined in [SP800-90A]) string consists of ≥ 256 bits that have been generated by a TRNG that is compliant with class PTG.2, PTG.3 or NTG.1.

Then, Hash-DRBG can be claimed to belong to the DRG.3 functionality class.

Note 58 [DRNG Seeding]	**CL2 and CL3**

The security of a DRNG derives from the proper seeding and reseeding of its internal state, which shall be done using a true random number generator that belongs to an agreed AIS-31 functionality class as described in the TRNG section of this methodology.

Note 59 [DRNG Backtracking Resistance]	CL2 and CL3
In systems that aim at providing perfect forward secrecy (PFS), an attacker who has recovered the current state of the random number generator which was used in previous key exchanges to produce ephemeral keys will be able to break the PFS property if it is possible to practically compute past DRNG outputs from the present state of the DRNG. Therefore, only DRNG which does not allow such backward computation should be used.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-AGREED/DRNGScheme	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The scheme utilized for the deterministic random number generation implemented by the TOE is considered as agreed upon according to Table 30. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned and that it complies with the specific notes. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned and that it complies with the specific notes.	**I1** **I2** **I6**

3.3.1.1.5.3 Random Number Generators with Specific Distributions

166. Many asymmetric cryptographic mechanisms require generating integers that follow a specific distribution. For such mechanisms, a random number generator cannot be used directly since it does not provide the required distribution directly. Therefore, random generators that offer specific output distributions must be implemented using random number generators.

3.3.1.1.5.3.1 Random Uniform Modular Distributions

167. In particular, asymmetric mechanisms that require using randomly generated integers that belong to a specific interval [0, q - 1], where q is usually not a power of 2, stand out. This is especially relevant when it comes to randomly generating an element within a finite group whose order is a prime power. In these cases, the authorized generators are known as modulo q uniform distribution generators.

168. In case the TOE implements random number generation following a uniform modular distribution:

- The **TOE shall** only implement agreed schemes for random uniform modular distributions according to the table below:

Agreed Schemes for Random Uniform Modular Distributions		
Scheme	**R/L**	**Notes**
Testing Technique [CCN-STIC-221] Algorithm 1	R	Note 60
Extra Random Technique [CCN-STIC-221] Algorithm 2	R	Note 60

Table 31. Agreed Schemes for Random Uniform Modular Distributions

Note 60 [DRNG as Input]	**CL2 and CL3**
Agreed Random Uniform Distribution schemes shall be applied to the output of an agreed deterministic random number generator. The schemes take a string of random bits (the output of an agreed DRNG) as input and output a random integer.	

3.3.1.1.5.3.2 Random Prime Number Generation

169. The generation of asymmetric parameters and RSA keys requires to generate random prime numbers. The generation of random primes adopts the following strategy: initially an integer of the appropriate size is drawn at random from a uniform distribution. Then, it is tested whether it satisfies some properties and a primality test is applied. As long as these tests fail, the candidate integer is updated and tested anew.

170. In case the TOE implements prime random number generation:

- The **TOE shall** only implement agreed methods for prime random number generation according to the table below and follow the specific notes:

Agreed Methods for Prime Random Generation		
Method	**R/L**	**Notes**
Method 1 [SOGIS-HEP] Append. B.1 [CCN-STIC-221] Algorithm 3	R	Note 61 Note 62 Note 63
Method 2 [SOGIS-HEP] Append. B.2 [CCN-STIC-221] Algorithm 4	R	Note 61 Note 62 Note 63

Table 32. Agreed Methods for Prime Random Generation

Note 61 [Primes from Uniform Distributions]	CL2 and CL3
Both prime number generation methods described above make use of agreed uniform modular distributions as described in section 3.3.1.1.5.3.1 "Random Uniform Modular Distributions" for the generation of the testing candidate *p.*	

Note 62 [Miller-Rabin as Primality Test]	CL2 and CL3
Method 1 and Method 2 for prime number generation entail the use of a primality test for the testing candidate which is described in the mentioned algorithms as *TestPrime(p)*. If the TOE implements prime random generation, the methods shall make use of the Miller-Rabin algorithm as described, for example, in Algorithm 5 of the [CCN-STIC-221] guidance for primality testing.	

Note 63 [Miller-Rabin Parameters]	CL2 and CL3
The Miller-Rabin algorithm shall be implemented according to the following conditions: ▪ In addition to the number p to be examined, the algorithm needs a random value $a \in \{2, 3, \ldots p-2\}$, the so-called basis. This parameter shall be generated from a uniform modular distribution [2, p-2] by an approved method (section 3.3.1.1.5.3.1 "Random Uniform Modular Distributions"). This condition shall be taken into account for each iteration. ▪ The number of iterations employed shall be large enough so that the probability that the testing candidate p is composite is lower than 2^{-125}.	

3.3.1.2 CRYPTOGRAPHIC PROTOCOLS IMPLEMENTATION

171. This second topic defines the cryptographic evaluation tests to verify that each cryptographic protocol and its underlying cryptographic mechanisms declared by the vendor in **I1** are considered agreed upon by CCN, in order to fulfill the **CCN-PROTOCOL** evaluation task.

3.3.1.2.1 Transport Layer Security Protocol (TLS)

172. The Transport Layer Security (TLS) protocol, previously known as the Secure Socket Layer (SSL) protocol, is a protocol that enables the protection of communications over the Internet. This includes secure connections such as those established through the Hypertext Transfer Protocol Secure (HTTPS) or the File Transfer Protocol Secure (FTPS), for instance. In essence, the TLS protocol

allows for the configuration and use of secure channels between the two parties in a connection: the client and the server.

173. In case the TOE implements the TLS cryptographic protocol to establish secure communications:

- The **TOE shall** only implement agreed TLS versions according to the table below:

Agreed TLS Protocol Versions		
Protocol	**R/L**	**Notes**
TLS v1.3 [RFC-8446]	R	
TLS v1.2 [RFC-5246]	R	

Table 33. Agreed TLS Protocol Versions

- The **tester shall** perform the following cryptographic evaluation test:

CCN-PROTOCOL/TLS	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The version used for the TLS protocol implemented by the TOE is considered as agreed upon according to Table 33. - The configuration used for the TLS v1.3 and TLS v1.2 protocol implemented by the TOE is considered as agreed upon. To be verified: ▪ Cipher suites ▪ Key Establishment Mechanisms ▪ Digital Signature Mechanisms - The TLS protocol configurations implemented by the TOE comply with all the associated specific notes. - All the underlying cryptographic mechanisms involved in the TLS protocol implemented by the TOE have been successfully evaluated in the CCN-AGREED evaluation task. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In addition, verify that the TLS protocol configuration implemented by the TOE matches with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned. In addition, verify that the TLS protocol configuration	**I1** **I2** **I6**

CCN-PROTOCOL/TLS	Inputs
implemented by the TOE matches with those declared by the vendor in **I1**.	

3.3.1.2.1.1 TLS v1.3

174. The convention followed for TLS v1.3 cipher suites is ***TLS_ENC_Long_Mod_Hash***, where *ENC* represents the encryption system, *Long* is the key length considered, *Mod* is the encryption mode, and *Hash* refers to the hash function considered.

175. In case the TOE implements the TLS v1.3 cryptographic protocol:

- The **TOE shall** only implement agreed TLS v1.3 cipher suites according to the table below:

Agreed TLS v1.3 Protocol Cipher Suites			
Code	**Cipher Suite**	**R/L**	**Notes**
0x1302	**TLS_AES_256_GCM_SHA384 [RFC-8446]**	R	
0x1301	**TLS_AES_128_GCM_SHA256 [RFC-8446]**	R	
0x1304	**TLS_AES_128_CCM_SHA256 [RFC-8446]**	R	
0x1303	**TLS_CHACHA20_POLY1305_SHA256 [RFC-7905]**	R	

Table 34. Agreed TLS v1.3 Protocol Cipher suites

176. In addition to the Diffie-Hellman key agreement over finite fields or elliptic curves, TLS v1.3 offers additional handshake protocol modes using Pre-Shared Keys (PSK). In this context, PSK refers to keys that are provided out of band or to keying material that has been established in a previous session through the session ticket mechanism.

177. In case the TOE implements the TLS v1.3 key establishment based on Pre-Shared Keys (PSK):

- The **TOE shall** only implement agreed PSK modes according to the table below and follow the specific notes:

Agreed TLS v1.3 Pre-Shared keys modes			
Code	**Mode**	**R/L**	**Notes**
0x0000	**psk_ke [RFC-8446]**	L[2026]	Note 64 Note 65

Agreed TLS v1.3 Pre-Shared keys modes			
Code	**Mode**	**R/L**	**Notes**
0x0001	**psk_dhe_ke [RFC-8446]**	R	Note 65

Table 35. Agreed TLS v1.3 PSK modes

Note 64 [psk_ke mode]	**CL1, CL2 and CL3**
The PSK mode "psk_ke" does not provide perfect persistent forward secrecy.	

Note 65 [0-RTT Data]	**CL1, CL2 and CL3**
TLS v1.3 offers an option to include application data in the first message of a PSK protocol, known as zero Round-Trip Time data or 0-RTT data. These data are not protected against replay attacks, so it is not recommended to send or accept this type of data.	

178. There are some extensions for the TLS v1.3 protocol that will be shown below, and they relate to the groups that can be used for key establishment, signature algorithms, etc.

    - The client and the server can use the *"supported_groups"* extension to inform each other about the Diffie-Hellman groups they wish to use for (EC)DHE.
    - The client and the server can utilize the *"signature_algorithms"* and *"signature_algorithms_cert"* extensions to mutually convey their desired signature algorithms for certificate-based authentication. The *"signature_algorithms"* extension pertains to signatures generated by the client or server for their *CertificateVerify* message, whereas the *"signature_algorithms_cert"* extension pertains to signatures on certificates.

179. In case the TOE implements DH or ECDH for the *"supported_groups"* extension to perform key establishment procedures in TLS v1.3 protocol:

- The **TOE shall** only implement agreed DH or ECDH groups according to the table below:

Agreed TLS v1.3 Diffie-Hellman Groups			
Code	**DH / ECDH Group**	**R/L**	**Notes**
0x0100	**ffdhe2048 [RFC-7919]**	L[2025]	

Agreed TLS v1.3 Diffie-Hellman Groups			
Code	**DH / ECDH Group**	**R/L**	**Notes**
0x0101	**ffdhe3072 [RFC-7919]**	R	
0x0102	**ffdhe4096 [RFC-7919]**	R	
0x0017	**P-256 (secp256r1) [RFC-8422]**	R	
0x0018	**P-384 (secp384r1) [RFC-8422]**	R	
0x0019	**P-521 (secp521r1) [RFC-8422]**	R	
0x001F	**BrainpoolP256r1tls13 [RFC-8734]**	R	
0x0020	**BrainpoolP384r1tls13 [RFC-8734]**	R	
0x0021	**BrainpoolP512r1tls13 [RFC-8734]**	R	
0x001D	**x25519 [RFC-8422] [RFC-7748]**	R	
0x001E	**x448 [RFC-8422] [RFC-7748]**	R	

Table 36. Agreed TLS v1.3 Diffie-Hellman Groups

180. In case the TOE implements digital signature cryptographic constructions for the *"signature_algorithms"* extension to perform client-server digital signature operations in TLS v1.3 protocol:

- The **TOE shall** only implement agreed digital signature schemes according to the table below:

Agreed TLS v1.3 Server-Client Digital Signature Schemes			
Code	**Signature Mechanism**	**R/L**	**Notes**
0x0804	**rsa_pss_rsae_sha256 [RFC-8446]**	R	
0x0805	**rsa_pss_rsae_sha384 [RFC-8446]**	R	
0x0806	**rsa_pss_rsae_sha512 [RFC-8446]**	R	

Agreed TLS v1.3 Server-Client Digital Signature Schemes			
Code	**Signature Mechanism**	**R/L**	**Notes**
0x0807	**ed25519 [RFC-8446]**	R	
0x0808	**ed448 [RFC-8446]**	R	
0x0809	**rsa_pss_pss_sha256 [RFC-8446]**	R	
0x080A	**rsa_pss_pss_sha384 [RFC-8446]**	R	
0x080B	**rsa_pss_pss_sha512 [RFC-8446]**	R	
0x0403	**ecdsa_secp256r1_sha256 [RFC-8446]**	R	
0x0503	**ecdsa_secp384r1_sha384 [RFC-8446]**	R	
0x0603	**ecdsa_secp521r1_sha512 [RFC-8446]**	R	
0x081A	**ecdsa_brainpoolP256r1tls13_sha256 [RFC-8734]**	R	
0x081B	**ecdsa_brainpoolP384r1tls13_sha384 [RFC-8734]**	R	
0x081C	**ecdsa_brainpoolP512r1tls13_sha512 [RFC-8734]**	R	

Table 37. Agreed TLS v1.3 Server-Client Digital Signature Schemes

181. In case the TOE implements digital signature cryptographic constructions for the *"signature_algorithms_cert"* extension to perform digital signature operations on certificates in TLS v1.3 protocol:

- The **TOE shall** only implement agreed digital signature schemes according to the table below:

Agreed TLS v1.3 Digital Signature Schemes on Certificates			
Code	**Signature Mechanism**	**R/L**	**Notes**
0x0401	**rsa_pkcs1_sha256 [RFC-8446]**	L[2025]	
0x0501	**rsa_pkcs1_sha384 [RFC-8446]**	L[2025]	

Agreed TLS v1.3 Digital Signature Schemes on Certificates			
Code	**Signature Mechanism**	**R/L**	**Notes**
0x0601	**rsa_pkcs1_sha512 [RFC-8446]**	L[2025]	
0x0804	**rsa_pss_rsae_sha256 [RFC-8446]**	R	
0x0805	**rsa_pss_rsae_sha384 [RFC-8446]**	R	
0x0806	**rsa_pss_rsae_sha512 [RFC-8446]**	R	
0x0807	**ed25519 [RFC-8446]**	R	
0x0808	**ed448 [RFC-8446]**	R	
0x0809	**rsa_pss_pss_sha256 [RFC-8446]**	R	
0x080A	**rsa_pss_pss_sha384 [RFC-8446]**	R	
0x080B	**rsa_pss_pss_sha512 [RFC-8446]**	R	
0x0403	**ecdsa_secp256r1_sha256 [RFC-8446]**	R	
0x0503	**ecdsa_secp384r1_sha384 [RFC-8446]**	R	
0x0603	**ecdsa_secp521r1_sha512 [RFC-8446]**	R	
0x081A	**ecdsa_brainpoolP256r1tls13_sha256 [RFC-8734]**	R	
0x081B	**ecdsa_brainpoolP384r1tls13_sha384 [RFC-8734]**	R	
0x081C	**ecdsa_brainpoolP512r1tls13_sha512 [RFC-8734]**	R	

Table 38. Agreed TLS v1.3 Digital Signature Algorithms on Certificates

3.3.1.2.1.2 TLS v1.2

182. In the TLS v1.2 version, the cryptographic mechanisms of a connection are defined by a cryptographic suite that specifies a key establishment mechanism

(with authentication) for the TLS Handshake protocol, an authenticated encryption mechanism for the TLS Record protocol, and a hash function for the key derivation process. Depending on the cipher suite, a group for the Diffie-Hellman key establishment must also be specified, either in a subgroup of a finite field or in an elliptic curve over a finite field, and a digital signature construction.

183. For these cipher suites, the following notation is typically used: ***TLS_AKE_WITH_ENC_Hash***, where *AKE* denotes a key establishment mechanism (with authentication), *ENC* is an encryption algorithm including its key length and mode of operation, and *Hash* refers to the hash function. The hash function is used for an HMAC that is used in a pseudo-random function (PRF), which is used for key derivation. In the case where the system under consideration uses the CCM mode of operation, no hash function is indicated and it is assumed that it uses SHA2-256 as the underlying hash function for the PRF. If *ENC* were not an authenticated encryption cryptographic construction, then HMAC is also used for integrity protection in the TLS Record protocol.

184. In case the TOE implements the TLS v1.2 cryptographic protocol:

- The **TOE shall** only implement agreed TLS v1.2 cipher suites according to the table below and follow the specific notes:

Agreed TLS v1.2 Protocol Cipher Suites			
Code	**Cipher Suite**	**R/L**	**Notes**
0xC02C	**TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 [RFC-5289]**	R	
0xC02B	**TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 [RFC-5289]**	R	
0xC0AD	**TLS_ECDHE_ECDSA_WITH_AES_256_CCM [RFC-7251]**	R	
0xC0AC	**TLS_ECDHE_ECDSA_WITH_AES_128_CCM [RFC-7251]**	R	
0xCCA9	**TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 [RFC-7905]**	R	
0xC030	**TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 [RFC-5289]**	L	
0xC02F	**TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 [RFC-5289]**	L	
0xCCA8	**TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 [RFC-7905]**	L	
0xC024	**TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 [RFC-5289]**	L[2025]	Note 66

Agreed TLS v1.2 Protocol Cipher Suites			
Code	**Cipher Suite**	**R/L**	**Notes**
0xC023	**TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 [RFC-5289]**	L[2025]	Note 66
0xC028	**TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 [RFC-5289]**	L[2025]	Note 66
0xC027	**TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 [RFC-5289]**	L[2025]	Note 66
0x009F	**TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 [RFC-5288]**	L	
0x009E	**TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 [RFC-5288]**	L	
0xC09F	**TLS_DHE_RSA_WITH_AES_256_CCM [RFC-6655]**	L	
0xC09E	**TLS_DHE_RSA_WITH_AES_128_CCM [RFC-6655]**	L	
0x006B	**TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 [RFC-5246]**	L[2025]	Note 66
0x0067	**TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 [RFC-5246]**	L[2025]	Note 66
0x009D	**TLS_RSA_WITH_AES_256_GCM_SHA384**	L	Note 67
0x009C	**TLS_RSA_WITH_AES_128_GCM_SHA256**	L	Note 67
0xC09D	**TLS_RSA_WITH_AES_256_CCM**	L	Note 67
0xC09C	**TLS_RSA_WITH_AES_128_CCM**	L	Note 67
0x003D	**TLS_RSA_WITH_AES_256_CBC_SHA256**	L[2025]	Note 66 Note 67
0x003C	**TLS_RSA_WITH_AES_128_CBC_SHA256**	L[2025]	Note 66 Note 67
0xCCAA	**TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 [RFC-7905]**	L	
0x006A	**TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 [RFC-5246]**	L	
0x00A3	**TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 [RFC-5288]**	L	

Table 39. Agreed TLS v1.2 Protocol Cipher Suites

Note 66 [TLS Encrypt and then MAC]	**CL1, CL2 and CL3**
TLS cryptographic suites whose encryption mechanisms are based on CBC shall be used in conjunction with the "*encrypt_then_mac*" extension.	

Note 67 [TLS with RSA]	**CL1, CL2 and CL3**
Key exchange with RSA does not provide Perfect Forward Secrecy (PFS).	

185. In case the TOE implements the TLS v1.2 cryptographic protocol using cipher suites with Pre-Shared Key (PSK) mode:

- The **TOE shall** only implement agreed TLS v1.2 cipher suites with Pre-Shared Key (PSK) mode according to the table below:

Agreed TLS v1.2 Protocol Cipher Suites with PSK			
Code	**Cipher Suite**	**R/L**	**Notes**
0xC037	**TLS_ECDHE_PSK_WITH_AES_128_CBC_ SHA256 [RFC-5489]**	R	
0xC038	**TLS_ECDHE_PSK_WITH_AES_256_CBC_ SHA384 [RFC-5489]**	R	
0xD001	**TLS_ECDHE_PSK_WITH_AES_128_GCM_ SHA256 [RFC-8442]**	R	
0xD002	**TLS_ECDHE_PSK_WITH_AES_256_GCM_ SHA384 [RFC-8442]**	R	
0xD005	**TLS_ECDHE_PSK_WITH_AES_128_CCM_ SHA256 [RFC-8442]**	R	
0xCCAD	**TLS_DHE_PSK_WITH_CHACHA20_ POLY1305_SHA256 [RFC-7905]**	R	
0x00B2	**TLS_DHE_PSK_WITH_AES_128_CBC_ SHA256 [RFC-5487]**	R	
0x00B3	**TLS_DHE_PSK_WITH_AES_256_CBC_ SHA384 [RFC-5487]**	R	
0x00AA	**TLS_DHE_PSK_WITH_AES_128_GCM_ SHA256 [RFC-5487]**	R	
0x00AB	**TLS_DHE_PSK_WITH_AES_256_GCM_ SHA384 [RFC-5487]**	R	
0xC0A6	**TLS_DHE_PSK_WITH_AES_128_CCM [RFC-6655]**	R	

Agreed TLS v1.2 Protocol Cipher Suites with PSK			
Code	**Cipher Suite**	**R/L**	**Notes**
0xC0A7	**TLS_DHE_PSK_WITH_AES_256_CCM [RFC-6655]**	R	

Table 40. Agreed TLS v1.2 Protocol Cipher Suites with PSK

186. In case the TOE implements DH or ECDH for the *"supported_groups"* extension to perform key establishment procedures in TLS v1.2 protocol:

- The **TOE shall** only implement agreed DH or ECDH groups according to the table below:

Agreed TLS v1.2 Diffie-Hellman Groups			
Code	**DH / ECDH Group**	**R/L**	**Notes**
0x0100	**ffdhe2048 [RFC-7919]**	L[2025]	
0x0101	**ffdhe3072 [RFC-7919]**	R	
0x0102	**ffdhe4096 [RFC-7919]**	R	
0x0017	**P-256 (secp256r1) [RFC-8422]**	R	
0x0018	**P-384 (secp384r1) [RFC-8422]**	R	
0x0019	**P-521 (secp521r1) [RFC-8422]**	R	
0x001F	**BrainpoolP256r1 [RFC-7027]**	R	
0x0020	**BrainpoolP384r1 [RFC-7027]**	R	
0x0021	**BrainpoolP512r1 [RFC-7027]**	R	
0x001D	**x25519 [RFC-8422] [RFC-7748]**	R	
0x001E	**x448 [RFC-8422] [RFC-7748]**	R	

Table 41. Agreed TLS v1.2 Diffie-Hellman Groups

187. In case the TOE implements digital signature cryptographic constructions for the *"signature_algorithms"* extension in TLS v1.2 protocol:

- The **TOE shall** only implement agreed digital signature schemes and hash functions according to the tables below:

Agreed TLS v1.2 Digital Signature Schemes			
Code	**Signature Mechanism**	**R/L**	**Notes**
0x0001	**rsa [RFC-5246]**	L[2025]	
0x0002	**dsa [RFC-5246]**	R	
0x0003	**ecdsa [RFC-5246]**	R	
0x0007	**ed25519 [RFC-8422]**	R	
0x0008	**ed448 [RFC-8422]**	R	

Table 42. Agreed TLS v1.2 Digital Signature Schemes

Agreed TLS v1.2 Hash Functions			
Code	**Hash Function**	**R/L**	**Notes**
0x0004	**SHA2-256 [RFC-5246]**	R	
0x0005	**SHA2-384 [RFC-5246]**	R	
0x0006	**SHA2-512 [RFC-5246]**	R	

Table 43. Agreed TLS v1.2 Hash Functions

3.3.1.2.2 Secure Shell (SSH)

188. SSH is a cryptographic security protocol originally designed to replace insecure remote connection protocols such as Telnet. This protocol can be used to establish a secure channel within an insecure network. The most common applications of the SSH protocol are logging into a remote system and running commands or applications on remote systems.

189. The SSH protocol consists of three subprotocols: Transport Layer Protocol, User Authentication Protocol, and Connection Protocol. The Transport Layer Protocol allows server authentication, encryption, integrity protection, and optionally data compression. It is based on the TCP/IP protocol. The User Authentication

Protocol is used to authenticate the user to the server and is based on the Transport Layer Protocol. Finally, the Connection Protocol is responsible for creating and managing logical channels within the encrypted tunnel and is based on the User Authentication Protocol.

190. In case the TOE implements the SSH cryptographic protocol to establish secure remote connections:

- The **TOE shall** only implement the agreed SSH version according to the table below:

Agreed SSH Protocol Version		
Protocol	**R/L**	**Notes**
SSH-2 [RFC-4251]	R	

Table 44. Agreed SSH Protocol Version

- The **tester shall** perform the following cryptographic evaluation test:

CCN-PROTOCOL/SSH	**Inputs**
In the case of **CL1** evaluations, verify in **I1** that: - The version used for the SSH protocol implemented by the TOE is considered as agreed upon according to Table 44. - The configuration used for the SSH protocol implemented by the TOE is considered as agreed upon. To be verified: ▪ Key Establishment Mechanisms ▪ Encryption Mechanisms ▪ Origin Integrity and Authenticity Mechanisms ▪ Server-Client Authentication Mechanisms - The SSH protocol configurations implemented by the TOE comply with all the associated specific notes. - All the underlying cryptographic mechanisms involved in the SSH protocol implemented by the TOE have been successfully evaluated in the CCN-AGREED evaluation task. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In addition, verify that the SSH protocol configuration implemented by the TOE matches with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned. In addition, verify that the SSH protocol configuration	**I1** **I2** **I6**

CCN-PROTOCOL/SSH	Inputs
implemented by the TOE matches with those declared by the vendor in **I1**.	

3.3.1.2.2.1 SSH Key Establishment

191. The key exchange mechanism of the SSH protocol is based on Diffie-Hellman. In fact, there are several recommended groups, all of them using the SHA2-512 function.

192. Regarding the DH or ECDH mechanisms implemented by the TOE for the key establishment of the SSH protocol:

- The **TOE shall** only implement agreed DH or ECDH groups according to the table below and follow the specific notes:

Agreed SSH Diffie-Hellman Groups		
DH / ECDH Group	**R/L**	**Notes**
DH-exchange-SHA256 [RFC-4419]	R	Note 68 Note 69
DH-group15-SHA512 [RFC-3526]	R	Note 69
DH-group16-SHA512 [RFC-3526]	R	Note 69
DH-group17-SHA512 [RFC-3526]	R	Note 69
DH-group18-SHA512 [RFC-3526]	R	Note 69
ECDH-SHA2-* [RFC-5656]	R	Note 69 Note 70

Table 45. Agreed SSH Diffie-Hellman Groups

Note 68 [Diffie-Hellman-group-exchange-SHA256]	CL2 and CL3
The size of the prime number to be used, p, should be at least 3000 bits. Additionally, the order of the generator should have a size of at least 2^{250}. Finally, p should be a safe prime. Remember that a safe prime, p, is a prime of the form $p = 1 + 2q$, where q is another prime. Therefore, the order of the generator g can only be a divisor of $p - 1 = 2 \cdot q$, which means it can either be 2 or q. This implies that the size of q should be much larger than the recommended minimum of 2^{250} for the order of g.	

Note 69 [Key Renewal]	CL2 and CL3
In order to further complicate attacks against session keys, it is recommended to renew the key used in a connection after a certain period or a certain amount of transmitted data. With SSH, this process can be initiated by both the client and the server by sending the *SSH_MSG_KEXINIT* message. Therefore, session keys should be renewed after one hour or after one gigabyte of data has been transmitted, whichever occurs first.	

Note 70 [ECDH-SHA2-*]	CL1, CL2 and CL3
The ECDH key exchange is defined by a family of method names. Each method name is the concatenation of the string "*ECDH-SHA2-" with* the domain parameter of the corresponding elliptic curve. The agreed-upon elliptic curves are as follows: secp256r1, secp384r1, and secp521r1. The agreed-upon hash function is the SHA2-512.	

3.3.1.2.2.2 SSH Encryption Mechanisms

193. Regarding the encryption and decryption mechanisms implemented by the TOE for the SSH protocol:

- The **TOE shall** only implement agreed encryption mechanisms according to the table below and follow the specific notes:

Agreed SSH Encryption Mechanisms		
Encryption Mechanism	**R/L**	**Notes**
AEAD_AES_128_GCM [RFC-5647]	R	Note 71
AEAD_AES_256_GCM [RFC-5647]	R	Note 71
AES128-CTR [RFC-4344]	R	Note 71
AES192-CTR [RFC-4344]	R	Note 71
AES256-CTR [RFC-4344]	R	Note 71

Table 46. Agreed SSH Encryption Mechanisms

Note 71 [Security Statement]	CL1, CL2 and CL3
To the extent possible, the AEAD_AES_128_GCM and AEAD_AES_256_GCM algorithms should be used because there are verifiable security statements for the GCM mode concerning the security goal of authenticated encryption. If unauthenticated encryption is chosen (i.e., AES-CTR), it will be mandatory to use one of the recommended options to protect the integrity and authenticity of messages (i.e., HMAC-SHA2).	

3.3.1.2.2.3 SSH Origin Integrity and Authenticity Mechanisms

194. Regarding the origin integrity and authenticity mechanisms implemented by the TOE for the SSH protocol:

- The **TOE shall** only implement agreed origin integrity and authenticity mechanisms according to the table below:

Agreed SSH Origin Integrity and Authenticity Mechanisms		
Scheme	**R/L**	**Notes**
HMAC-SHA1 [RFC-4253]	L[2030]	
HMAC-SHA2-256 [RFC-4344]	R	
HMAC-SHA2-512 [RFC-4344]	R	
AEAD_AES_128_GCM [RFC-5647]	R	
AEAD_AES_256_GCM [RFC-5647]	R	

Table 47. Agreed SSH Origin Integrity and Authenticity Mechanisms

3.3.1.2.2.4 SSH Server and Client Authentication Mechanisms

195. Regarding server-client authentication mechanisms implemented by the TOE for the SSH protocol:

- The **TOE shall** only implement agreed server-client authentication mechanisms according to the table below and follow the specific notes:

Agreed SSH Server-Client Authentication Mechanisms		
Scheme	**R/L**	**Notes**
ECDSA-SHA2-* [RFC-5656]	R	Note 72

Agreed SSH Server-Client Authentication Mechanisms		
Scheme	**R/L**	**Notes**
x509v3-ECDSA-SHA2-* **[RFC-6187]**	R	Note 73

Table 48. Agreed SSH Server-Client Authentication Mechanisms

Note 72 [ECDSA-SHA2-*]	**CL1, CL2 and CL3**
The ECDSA scheme is defined by a family of method names. Each method name is the concatenation of the string *"ECDSA-SHA2-"* with the domain parameter of the corresponding elliptic curve. The agreed-upon elliptic curves are as follows: secp256r1, secp384r1, and secp521r1.	

Note 73 [x509v3-ECDSA-SHA2-*]	**CL1, CL2 and CL3**
The x509v3-ECDSA scheme is defined by a family of method names. Each method name is the concatenation of the string "*x509v3-ECDSA-SHA2-*" with the domain parameter of the corresponding elliptic curve. The agreed-upon elliptic curves are as follows: secp256r1, secp384r1, and secp521r1.	

3.3.1.2.3 Internet Protocol Security (IPsec) with IKEv2

196. The most important use of IPsec is to create Virtual Private Networks or VPNs, which means establishing secure communication channels over insecure IP networks. IPsec is a standard that provides security at the network layer of the Internet Protocol (IP) in the TCP/IP protocol stack (Transmission Control Protocol/Internet Protocol). Unlike TLS and SSH protocols, IPsec provides security at higher layers, including the application layer.

197. In case the TOE implements the IPsec cryptographic protocol to establish secure remote connections:

- The **TOE shall** only implement agreed IPsec versions according to the table below and follow the specific notes:

Agreed IPsec Protocol Versions		
Protocol	**R/L**	**Notes**
IPsec **[RFC-8221]**	R	Note 74 Note 75
IKEv2 **[RFC-7296] [RFC-8247]**	R	Note 74
ESP **[RFC-4303]**	R	Note 74 Note 75

Table 49. Agreed IPsec Protocol Versions

Note 74 [Tunnel Mode – Transport Mode]	**CL2 and CL3**
IPsec can be deployed in tunnel mode and transport mode. In tunnel mode, the entire IP packet is cryptographically protected, while in transport mode, the original IP packet header is preserved, and some security fields are added. It can be stated that tunnel mode provides more security than transport mode, so the former should be used before the latter. Transport mode is justified only if packet size becomes problematic due to network-imposed restrictions.	

Note 75 [IPsec Attacks]	**CL1, CL2 and CL3**
There are known attacks when IPsec is implemented in any MAC-then-Encrypt configuration (such as using AH in transport mode before ESP with only encryption in tunnel mode). However, no attacks are known against IPsec when using ESP with only encryption followed by AH or ESP with authenticity and integrity protection without another AH following it. Therefore, it is recommended to use ESP with integrity and authenticity protection options in addition to confidentiality.	

- The **tester shall** perform the following cryptographic evaluation test:

CCN-PROTOCOL/IPsec	Inputs
In the case of **CL1** evaluations, verify in **I1** that: - The versions used for the IPsec protocol implemented by the TOE are considered as agreed upon according to Table 49. - The configuration used for the IPsec protocol implemented by the TOE is considered as agreed upon. To be verified: ▪ IKEv2 Key Establishment Mechanisms ▪ IKEv2 and ESP Encryption Mechanisms ▪ IKEv2 and ESP Origin Integrity and Authenticity Mechanisms ▪ IKEv2 and ESP Pseudo-Random Functions ▪ IKEv2 Authentication Mechanisms - The IPsec protocol configurations implemented by the TOE comply with all the associated specific notes. - All the underlying cryptographic mechanisms involved in the IPsec protocol implemented by the TOE have been successfully evaluated in the CCN-AGREED evaluation task. In the case of **CL2** evaluations, operate the TOE (using **I2**) to verify the above-mentioned. In addition, verify that the IPsec protocol configuration implemented by the TOE matches with those declared by the vendor in **I1**. In the case of **CL3** evaluations, operate the TOE (using **I2**) and perform a source code review (using **I6**) to verify the above-mentioned. In addition, verify that the IPsec protocol configuration implemented by the TOE matches with those declared by the vendor in **I1**.	**I1** **I2** **I6**

3.3.1.2.3.1 IKEv2 Key Establishment

198. The key exchange mechanism used in the IKEv2 protocol is based on Diffie-Hellman.

199. Regarding the DH or ECDH mechanisms implemented by the TOE for the key establishment of the IKEv2 protocol:

- The **TOE shall** only implement agreed DH or ECDH groups according to the table below:

Agreed IKEv2 Diffie-Hellman Groups		
DH / ECDH Group	**R/L**	**Notes**
3072-bit MODP [RFC-3526]	R	
4096-bit MODP [RFC-3526]	R	
6144-bit MODP [RFC-3526]	R	
8192-bit MODP [RFC-3526]	R	
256-bit random ECP [RFC-5903]	R	
384-bit random ECP [RFC-5903]	R	
521-bit random ECP [RFC-5903]	R	
brainpoolP256r1 [RFC-6954]	R	
brainpoolP384r1 [RFC-6954]	R	
brainpoolP512r1 [RFC-6954]	R	
x25519 [RFC-8031]	R	
x448 [RFC-8031]	R	

Table 50. Agreed IKEv2 Diffie-Hellman Groups

3.3.1.2.3.2 IKEv2 and ESP Encryption Mechanisms

200. Regarding the encryption and decryption mechanisms implemented by the TOE for the IKEv2 and ESP protocols:

- The **TOE shall** only implement agreed symmetric encryption cryptographic constructions (according to Table 12) and agreed authenticated encryption cryptographic constructions (according to Table 16).

3.3.1.2.3.3 IKEv2 and ESP Origin Integrity and Authenticity Mechanisms

201. Regarding the origin integrity and authenticity mechanisms implemented by the TOE for the IKEv2 and ESP protocols:

- The **TOE shall** only implement agreed origin integrity and authenticity mechanisms according to the table below and follow the specific notes:

Agreed IKEv2 and ESP Origin Integrity and Authenticity Mechanisms		
MAC Construction	**R/L**	**Notes**
HMAC-SHA2-256_128 [RFC-4868]	R	Note 76
HMAC-SHA2-384_192 [RFC-4868]	R	Note 76
HMAC-SHA2-512_256 [RFC-4868]	R	Note 76
AES-CMAC-96 [RFC-4494]	R	
AES-GMAC [RFC-4543]	R	

Table 51. Agreed IKEv2 and ESP Origin Integrity and Authenticity Mechanisms

Note 76 [HMAC-SHA-XXX-YYY]	**CL1, CL2 and CL3**
Each of these schemes uses a fixed key length of XXX bits, truncating the output to YYY bits.	

3.3.1.2.3.4 IKEv2 Pseudo-Random Functions

202. Regarding the pseudo-random functions implemented by the TOE for the IKEv2 protocol key derivation:

- The **TOE shall** only implement agreed pseudo-random functions according to the table below:

Agreed IKEv2 Pseudo-Random Functions		
MAC Schemes	**R/L**	**Notes**
HMAC-SHA2-256 [RFC-4868]	R	
HMAC-SHA2-384 [RFC-4868]	R	
HMAC-SHA2-512 [RFC-4868]	R	
AES128-CMAC [RFC-4615]	R	

Table 52. Agreed IKEv2 Pseudo-Random Functions

3.3.1.2.3.5 IKEv2 Authentication Mechanisms

203. The cryptographic authentication mechanism for IKEv2 is the digital signature, recommending the use of X.509 certificates.

204. Regarding the authentication mechanisms implemented by the TOE to perform digital signature operations for the IKEv2 protocol:

- The **TOE shall** only implement agreed digital signature cryptographic constructions according to the table below and follow the specific notes:

Agreed IKEv2 Digital Signature Schemes		
Signature Mechanism	**R/L**	**Notes**
RSA (RSASSA-PSS) [RFC-4055]	R	Note 77
ECDSA-SHA-256 and P-256 [RFC-4754]	R	
ECDSA-SHA-384 and P-384 [RFC-4754]	R	
ECDSA-SHA-512 and P-521 [RFC-4754]	R	
ECDSA-256-BrainpoolP256r1 [RFC-7427]	R	
ECDSA-384-BrainpoolP384r1 [RFC-7427]	R	
ECDSA-512-BrainpoolP512r1 [RFC-7427]	R	
ECGDSA-256-BrainpoolP256r1 [RFC-7427]	R	
ECGDSA-384-BrainpoolP384r1 [RFC-7427]	R	
ECGDSA-512-BrainpoolP512r1 [RFC-7427]	R	

Table 53. Agreed IKEv2 Digital Signature Schemes

Note 77 [RSASSA-PSS]	**CL1, CL2 and CL3**
This scheme should only be used with PSS and with a SHA-2 family function.	

4 CONFORMITY TESTING

4.1 OBJECTIVE

205. This chapter defines the conformity testing process for the cryptographic mechanisms implemented by the TOE to verify the correct operation.

206. Therefore, it provides vendors with the conformity testing requirements that shall be met by the TOE to perform a correct operation of the cryptographic mechanisms. In addition, it provides testers with the evaluation task necessary to verify the correct execution and implementation of the cryptographic mechanisms during the evaluation process.

4.2 DEFINITIONS

207. The conformity testing process is based on the use of test vectors that will be used as input to stimulate the operation of the cryptographic mechanisms implemented in the TOE to obtain their output which will be compared with the output obtained by the reference implementation to determine the correct operation of each cryptographic mechanism and for each parameterization independently.

208. To thoroughly evaluate the functionality of cryptographic implementations, three types of conformity tests are utilized. These tests provide a comprehensive assessment of the implementation's behavior, uncovering any potential weaknesses or deviations from expected results.

Monte Carlo Tests (MCT)

209. The Monte Carlo Test consists of iteratively applying the cryptographic function, starting from a predefined seed or a randomly generated seed, to test against a large number of pseudo-random values. In the case of a predefined seed, the final result is compared to a reference value. When using a randomly generated seed, the same Monte Carlo Test is run on a reference implementation (using the same seed), and the final results of both are then compared.

210. Monte Carlo Tests are designed to thoroughly test cryptographic primitives, including block ciphers, fixed-length input hash functions, modular exponentiations, EC scalar multiplications, multi-scalar multiplications, etc.

211. To do so, the cryptographic function f is applied to values that are updated by encoding functions based on previous outputs of f.

212. A pseudocode description of the Monte Carlo mechanisms is provided in "Annex B: Conformity Tests Specification".

Conformity Known Answer Tests (KATs)

213. The Known Answer Test consists of applying a cryptographic mechanism to a fixed set of inputs and comparing the output to the expected results. While MCTs are useful for intensively testing a primitive, conformity KATs are designed to

complement them by allowing for more specific testing of a cryptographic mechanism's behavior.

214. Depending on the type of mechanism, different types of conformity Known Answer Tests are used:

   a) **Non-Specific KATs:** conformity KATs can include non-specific inputs to provide a rough check of the implementation's correct performance, which is especially useful for complex mechanisms like asymmetric constructions that rely on multiple other primitives.

   b) **Length Tests:** length tests are relevant for mechanisms with variable input or output lengths, such as hash functions or MAC with truncation. These tests verify the implementation's chaining mechanism and padding implementation and ensure that limitations on the range of input length are being enforced.

   c) **Corner Cases:** while MCTs, non-specific conformity KATs, and length tests all allow for testing the normal behavior of a mechanism, some corner cases may not be triggered by chance. These corner cases, including padding verification, can be tested through the use of specific inputs designed to trigger them.

Validation Tests (VAL)

215. Validation Tests consist of the generation of a certain number of test vectors by the vendor and their subsequent verification by the tester. This allows verifying the correct generation of asymmetric key pairs, the generation of digital signatures, and the key establishment process.

4.3 CONFORMITY TESTING PROCESS

Generation of Test Vectors

216. In the first step, the **tester shall** generate a request file (REQ) for each cryptographic mechanism implemented in the TOE, containing the test vectors associated with the supported configuration. Additionally, the **tester shall** generate a sample file (SAM) for each cryptographic mechanism implemented in the TOE, containing an example solution to indicate the format of the expected result.

217. The **tester shall** send the vendor a file package containing the REQ and SAM files associated with all the cryptographic mechanisms implemented in the TOE.

Generation of TOE Results

218. In the second step, the **vendor shall** implement a test harness (**I3**) to obtain the response files for all the cryptographic mechanisms implemented in the TOE (**I4**), as part of the conformity testing process.

219. The **test harness shall** receive as input the REQ file associated with a cryptographic mechanism and **shall** generate an RSP output file containing the

result of the TOE cryptographic operation. The **test harness shall** maintain the format presented in the REQ and SAM files for the generation of the RSP file.

220. Using this test harness (**I3**)**,** the **tester shall** generate a response file (RSP) associated with each cryptographic mechanism implemented, which shall contain the output provided by the TOE for each of the test vectors provided in the REQ file.

221. The **tester shall** ensure that the set of methods or functions defined in the test harness (**I3**) is complete and performs the appropriate calls to the TOE.

Generation of Results

222. In the third step, the **tester shall** generate the RSP file associated with each cryptographic mechanism implemented in the TOE, using the CCN Cryptographic Tool as the cryptographic reference implementation.

223. The **tester shall** maintain the format presented in the REQ and SAM files for the generation of the RSP file.

Validation of Results

224. In the fourth step, the **tester shall** validate the RSP files for each cryptographic mechanism implemented in the TOE, comparing the results provided with those obtained in the previous phase using the CCN Cryptographic Tool.

225. The **tester shall** determine if the TOE correctly implements the cryptographic constructions and primitives used and declared.

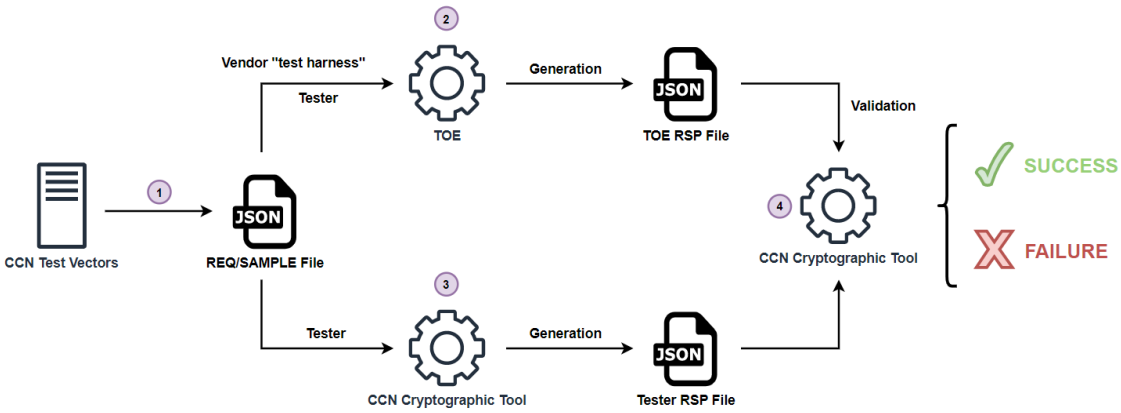

Figure 1. Conformity Testing Process Flow Diagram

4.4 CRYPTOGRAPHIC EVALUATION TASK

226. This section contains the conformity testing process for all the cryptographic mechanisms implemented by the TOE that shall be performed by the tester in order to verify their correct operation.

227. A cryptographic evaluation task related to the conformity testing procedure of the cryptographic mechanisms implemented has been defined:

CRYPTOGRAPHIC EVALUATION TASK	CCN-CONFORMITY	CERTIFICATION LEVEL
The **tester shall** perform the conformity testing process to verify the correct operation of all the cryptographic mechanisms implemented by the TOE. **Note:** In the case of **CL2** and **CL3** evaluations, all implemented cryptographic mechanisms shall be verified, even if they are not in use.		**CL1, CL2 and CL3**

228. For the correct completion of the entire evaluation task, it has been assigned to a cryptographic evaluation test that the **tester shall** perform for all the cryptographic mechanisms implemented by the TOE.

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
CCN-CONFORMITY **Cryptographic Mechanisms Conformity Testing**	CCN-CONFORMITY/Mechanisms	Mandatory

Table 54. Cryptographic Evaluation Test defined for CCN Conformity

4.4.1 CRYPTOGRAPHIC EVALUATION TEST DEFINITION

229. This subsection defines the cryptographic evaluation test that shall be performed by the tester to accomplish the evaluation task, based on the conformity testing process for each of the cryptographic mechanisms implemented by the TOE.

4.4.1.1 CRYPTOGRAPHIC MECHANISMS IMPLEMENTATION

230. This topic defines the cryptographic evaluation test to verify the correct operation of each cryptographic mechanism declared by the vendor in **I1**, in order to fulfill the **CCN-CONFORMITY** evaluation task.

231. For each construction and cryptographic primitive implemented:

- The **vendor shall** provide a testing interface (test harness) of the cryptographic mechanisms (**I3**) to perform the conformity testing process using the TOE cryptographic implementation.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-CONFORMITY/Mechanisms	Inputs
For **CL1**, **CL2**, and **CL3** evaluations, verify the correct operation of all the cryptographic mechanisms implemented by the TOE and defined by the vendor in **I1**. For this purpose: - Perform the conformity testing process of all the cryptographic mechanisms using **I3** to generate the TOE results files (**I4**). - Validate the TOE results files (**I4**) using the CCN Cryptographic Tool as the cryptographic reference implementation. **Note:** In the case of conformity testing on cryptographic constructions, it is also mandatory to perform the conformity testing process of the underlying cryptographic primitives.	**I1** **I3** **I4**

232. In the following, a set of tables is defined to identify the required conformity tests to be performed for each cryptographic construction. They are organized by constructions rather than by primitives because primitives are not used independently by the TOE (except for hash and XOF primitives) but as part of a cryptographic construction. Each conformity test corresponds to a unique ID. Details of the properties of each test are given in "Annex B: Conformity Tests Specification".

4.4.1.1.1 Symmetric Cryptographic Mechanisms

4.4.1.1.1.1 Hash Functions

233. For hashing cryptographic primitives, the following conformity tests have been defined:

Hash Functions Conformity Testing	
Primitive	**Conformity Test ID**
SHA-2	Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256*
SHA-3	Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256*

Hash Functions Conformity Testing	
Primitive	**Conformity Test ID**
	KAT_SHA3.2 for SHA3-384 *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1*
BLAKE2b	Primitive (BLAKE2b) *MCT_BLAKE2b.1* *KAT_BLAKE2b.1*

Table 55. Hash Functions Conformity Testing

4.4.1.1.1.2 Extendable-Output Functions (XOF)

234. For XOFs, the following conformity tests have been defined:

XOF Conformity Testing	
Primitive	**Conformity Test ID**
SHAKE	Primitive (SHAKE) *KAT_SHAKE.1 for SHAKE128* *KAT_SHAKE.2 for SHAKE256* Underlying Primitive (Keccak) *MCT_KECCAK.1*
cSHAKE	Primitive (cSHAKE) *KAT_CSHAKE.1 for cSHAKE128* *KAT_CSHAKE.2 for cSHAKE256* Underlying Primitive (SHAKE) *KAT_SHAKE.1 for cSHAKE128* *KAT_SHAKE.2 for cSHAKE256* Underlying Primitive (Keccak) *MCT_KECCAK.1*

Table 56. XOF Conformity Testing

4.4.1.1.1.3 Symmetric Encryption (Confidentiality Only)

235. For symmetric encryption cryptographic constructions that provide only confidentiality to the processed information, the following conformity tests have been defined:

Symmetric Encryption (Confidentiality Only) Conformity Testing		
Underlying	**Operation Mode**	**Conformity Test ID**
AES	**CBC**	Construction (AES-CBC) *KAT_CBC.1 for No-Padding* *KAT_CBC.2 for ISO-7816 Padding* *KAT_CBC.3 for PKCS#7 Padding* Underlying Primitive (AES) *MCT_AES.1*
	CBC-CS	Construction (AES-CBC-CS) *KAT_CBC-CS.1* Underlying Primitive (AES) *MCT_AES.1*
	CFB	Construction (AES-CFB) *KAT_CFB.1* Underlying Primitive (AES) *MCT_AES.1*
	CTR	Construction (AES-CTR) *KAT_CTR.1* Underlying Primitive (AES) *MCT_AES.1*
	OFB	Construction (AES-OFB) *KAT_OFB.1* Underlying Primitive (AES) *MCT_AES.1*

Table 57. Symmetric Encryption Conformity Testing

4.4.1.1.1.4 Disk Encryption

236. For symmetric encryption cryptographic constructions that perform disk encryption operations, the following conformity tests have been defined:

Disk Encryption Conformity Testing		
Underlying	**Operation Mode**	**Conformity Test ID**
AES	**XTS**	Construction (AES-XTS) *KAT_XTS.1 for XTS-256* *KAT_XTS.2 for XTS-512* Underlying Primitive (AES) *MCT_AES.1*

Table 58. Disk Encryption Conformity Testing

4.4.1.1.1.5 Integrity Modes: Message Authentication Code

237. For MAC cryptographic constructions, the following conformity tests have been defined:

MAC Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
AES	**CMAC**	Construction (AES-CMAC) *KAT_CMAC.1* Underlying Primitive (AES) *MCT_AES.1*
	CBC-MAC	Construction (AES-CBC-MAC) *KAT_CBC-MAC.1* Underlying Primitive (AES) *MCT_AES.1*
SHA-2	**HMAC**	Construction (HMAC) *KAT_HMAC.1 for all except SHA2-512/256* *KAT_HMAC.2 for SHA2-512/256* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256*
SHA-3	**HMAC**	Construction (HMAC) *KAT_HMAC.1* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1*
SHA-1	**HMAC-SHA1**	Construction (HMAC) *KAT_HMAC-1.1*

MAC Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
AES	**GMAC**	Construction (AES-GMAC) *KAT_GMAC.1* Underlying Primitive (AES) *MCT_AES.1*
cSHAKE	**KMAC-128**	Construction (KMAC) *KAT_KMAC.1* Underlying Primitive (cSHAKE128) *KAT_CSHAKE.1* Underlying Primitive (SHAKE128) *KAT_SHAKE.1* Underlying Primitive (Keccak) *MCT_KECCAK.1*
	KMAC-256	Construction (KMAC) *KAT_KMAC.1* Underlying Primitive (cSHAKE256) *KAT_CSHAKE.2* Underlying Primitive (SHAKE256) *KAT_SHAKE.2* Underlying Primitive (Keccak) *MCT_KECCAK.1*

Table 59. MAC Conformity Testing

4.4.1.1.1.6 Symmetric Entity Authentication Schemes

238. For symmetric entity authentication constructions, the following conformity tests have been defined:

Symmetric Entity Authentication Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
AES mode or MAC	**Symmetric Entity Authentication**	Underlying Construction (AES) *See Table 57* Underlying Construction (MAC) *See Table 59*

Table 60. Symmetric Entity Authentication Conformity Testing

4.4.1.1.1.7 Authenticated Encryption (AE)

239. For authenticated encryption cryptographic constructions, the following conformity tests have been defined:

Authenticated Encryption Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
AES mode + MAC (construction + construction)	**Encrypt-then-MAC**	Underlying Construction (AES) *See Table 57* Underlying Construction (MAC) *See Table 59*
AES	**CCM**	Construction (AES-CCM) *KAT_CCM.1* Underlying Primitive (AES) *MCT_AES.1*
	GCM	Construction (AES-GCM) *KAT_GCM.1 for 128-bit key* *KAT_GCM.2 for 192-bit key* *KAT_GCM.3 for 256-bit key* Underlying Primitive (AES) *MCT_AES.1*
	EAX	Construction (AES-EAX) *KAT_EAX.1 for 128-bit key* *KAT_EAX.2 for 192-bit or 256-bit keys* Underlying Primitive (AES) *MCT_AES.1*
ChaCha20 + Poly1305 (primitive + construction)	**ChaCha20-Poly1305**	Construction (ChaCha20-Poly1305) *KAT_CHACHA20-POLY1305.1* Underlying Construction (Poly1305) *MCT_POLY1305.1* *KAT_POLY1305.1* Underlying Primitive (ChaCha20) *MCT_CHACHA20.1* *KAT_CHACHA20.1*

Table 61. Authenticated Encryption Conformity Testing

4.4.1.1.1.8 Key Protection

240. For key protection cryptographic constructions, the following conformity tests have been defined:

Key Protection Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
AES-CMAC AES-CTR	**SIV**	Construction (AES-SIV) *KAT_SIV.1 for SIV-256* *KAT_SIV.2 for SIV-384* *KAT_SIV.3 for SIV-512* Underlying Construction (AES-CMAC) *KAT_CMAC.1* Underlying Construction (AES-CTR) *KAT_CTR.1* Underlying Primitive (AES) *MCT_AES.1*
AES	**KW**	Construction (AES-KW) *KAT_KW.1 for 128-bit key* *KAT_KW.2 for 192-bit key* *KAT_KW.3 for 256-bit key* Underlying Primitive (AES) *MCT_AES.1*
AES	**KWP**	Construction (AES-KWP) *KAT_KWP.1 for 128-bit key* *KAT_KWP.2 for 192-bit key* *KAT_KWP.3 for 256-bit key* Underlying Primitive (AES) *MCT_AES.1*

Table 62. Key Protection Conformity Testing

Note: In case the TOE implements key protection based on Authenticated Encryption cryptographic mechanisms, the **tester shall** execute the conformity tests defined in section 4.4.1.1.1.7 "Authenticated Encryption (AE)" depending on the implemented mechanisms.

4.4.1.1.1.9 Key Derivation Functions (KDF)

241. For key derivation cryptographic constructions, the following conformity tests have been defined:

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
HMAC	**NIST SP800-56 A/B/C OneStep**	Construction (SP800-56C KDF) *KAT_SP80056C-KDF.1* Underlying Construction (HMAC) *KAT_HMAC.1 for all except SHA2-512/256 and SHA-1* *KAT_HMAC.2 for SHA2-512/256* *KAT_HMAC-1.1 for SHA-1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
KMAC	**NIST SP800-56 A/B/C OneStep**	Construction (SP800-56C KDF) *KAT_SP80056C-KDF.2* Underlying Construction (KMAC) *KAT_KMAC.1* Underlying Primitive (cSHAKE) *KAT_CSHAKE.1 for KMAC-128* *KAT_CSHAKE.2 for KMAC-256* Underlying Primitive (SHAKE) *KAT_SHAKE.1 for KMAC-128* *KAT_SHAKE.2 for KMAC-256* Underlying Primitive (Keccak) *MCT_KECCAK.1*

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
SHA	**NIST SP800-56 A/B/C OneStep**	Construction (SP800-56C KDF) *KAT_SP80056C-KDF.3* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
HMAC NIST SP800-108	**NIST SP800-56 A/B/C TwoStep**	Construction (SP800-56C KDF) *KAT_SP80056C-KDF.4* Underlying Construction (HMAC) *KAT_HMAC.1 for all except SHA2-512/256 and SHA-1* *KAT_HMAC.2 for SHA2-512/256* *KAT_HMAC-1.1 for SHA-1* Underlying Construction (SP800-108) *KAT_SP800108-KDF.2 for NIST SP800-108 KDF in Counter mode* *KAT_SP800108-KDF.3 for NIST SP800-108 KDF in Feedback mode* *KAT_SP800108-KDF.4 for NIST SP800-108 KDF in Double Pipeline mode* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
CMAC NIST SP800-108	**NIST SP800-56 A/B/C TwoStep**	Construction (SP800-56C KDF) *KAT_SP80056C-KDF.5* Underlying Construction (AES-CMAC) *KAT_CMAC.1* Underlying Construction (SP800-108) *KAT_SP800108-KDF.2 for NIST SP800-108 KDF in Counter mode* *KAT_SP800108-KDF.3 for NIST SP800-108 KDF in Feedback mode* *KAT_SP800108-KDF.4 for NIST SP800-108 KDF in Double Pipeline mode* Underlying Primitive (AES) *MCT_AES.1*

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
HMAC	**NIST SP800-108**	Construction (SP800-108) *KAT_SP800108-KDF.2 for Counter mode* *KAT_SP800108-KDF.3 for Feedback mode* *KAT_SP800108-KDF.4 for Double Pipeline mode* Underlying Construction (HMAC) *KAT_HMAC.1 for all except SHA2-512/256 and SHA-1* *KAT_HMAC.2 for SHA2-512/256* *KAT_HMAC-1.1 for SHA-1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
CMAC	**NIST SP800-108**	Construction (SP800-108) *KAT_SP800108-KDF.2 for Counter mode* *KAT_SP800108-KDF.3 for Feedback mode* *KAT_SP800108-KDF.4 for Double Pipeline mode* Underlying Construction (AES-CMAC) *KAT_CMAC.1* Underlying Primitive (AES) *MCT_AES.1*
KMAC	**NIST SP800-108**	Construction (SP800-108) *KAT_SP800108-KDF.1 for KMAC* *KAT_SP800108-KDF.2 for Counter mode* *KAT_SP800108-KDF.3 for Feedback mode* *KAT_SP800108-KDF.4 for Double Pipeline mode* Underlying Construction (KMAC) *KAT_KMAC.1* Underlying Primitive (cSHAKE) *KAT_CSHAKE.1 for KMAC-128* *KAT_CSHAKE.2 for KMAC-256* Underlying Primitive (SHAKE) *KAT_SHAKE.1 for KMAC-128* *KAT_SHAKE.2 for KMAC-256* Underlying Primitive (Keccak) *MCT_KECCAK.1*

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
SHA	**ANSI-X9.63 KDF**	Construction (ANSI-X9.63 KDF) *KAT_X9.63-KDF.1 for all except SHA2-512 and SHA3-512* *KAT_X9.63-KDF.2 for SHA2-512 and SHA3-512* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
HMAC	**PBKDF2**	Construction (PBKDF2) *KAT_PBKDF2.1* Underlying Construction (HMAC) *KAT_HMAC.1 for all except SHA2-512/256 and SHA-1* *KAT_HMAC.2 for SHA2-512/256* *KAT_HMAC-1.1 for SHA-1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256*

Key Derivation Functions Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
HMAC	**HKDF**	Construction (HKDF) *KAT_HKDF.1 for SHA-1* *KAT_HKDF.2 for SHA2-256* *KAT_HKDF.3 for SHA2-384* *KAT_HKDF.4 for SHA2-512* Underlying Construction (HMAC) *KAT_HMAC.1 for SHA-2 family* *KAT_HMAC-1.1 for SHA-1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256*

Table 63. Key Derivation Functions Conformity Testing

4.4.1.1.1.10 Password Protection/Hashing Mechanisms

242. For password protection/hashing cryptographic constructions, the following conformity tests have been defined:

Password Protection/Hashing Mechanisms Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
BLAKE2b	**Argon2id**	Construction (Argon2id) *KAT_Argon2id.1* Underlying Primitive (BLAKE2b) *MCT_BLAKE2b.1* *KAT_BLAKE2b.1*
HMAC	**PBKDF2**	Construction (PBKDF2) *KAT_PBKDF2.1* Underlying Construction (HMAC) *KAT_HMAC.1 for all except SHA2-512/256 and SHA-1* *KAT_HMAC.2 for SHA2-512/256* *KAT_HMAC-1.1 for SHA-1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256*
PBKDF2	**SCRYPT**	Construction (SCRYPT) *KAT_SCRYPT.1* Underlying Construction (PBKDF2) *KAT_PBKDF2.1* Underlying Construction (HMAC) *KAT_HMAC.1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1*

Table 64. Password Protection/Hashing Mechanisms Conformity Testing

4.4.1.1.2 Asymmetric Cryptographic Mechanisms

4.4.1.1.2.1 Asymmetric Encryption Schemes

243. For asymmetric encryption cryptographic constructions, the following conformity tests have been defined:

Asymmetric Encryption Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
RSA SHA	**OAEP (PKCS#1v2.1)**	Construction (RSA-OAEP) *KAT_OAEP.1* Underlying Primitive (RSA) *MCT_RSA.1 for modular exponentiation operations with an RSA public key* *MCT_RSA.2 for modular exponentiation operations with an RSA public key, when the value e is equal to 65537* *MCT_RSA.3 for modular exponentiation operations with an RSA private key* *MCT_RSA.4 for modular exponentiation operations with extended parameters with a Chinese Remainder Theorem (CRT) recombination* *KAT_M-R.1 for Miller-Rabin in case of RSA Key Generation* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
RSA	**PKCS#1v1.5**	Construction (RSA-PKCS#1v1.5) *KAT_PKCS.1* Underlying Primitive (RSA)

		MCT_RSA.1 for modular exponentiation operations with an RSA public key *MCT_RSA.2 for modular exponentiation operations with an RSA public key, when the value e is equal to 65537* *MCT_RSA.3 for modular exponentiation operations with an RSA private key* *MCT_RSA.4 for modular exponentiation operations with extended parameters with a Chinese Remainder Theorem (CRT) recombination* *KAT_M-R.1 for Miller-Rabin in case of RSA Key Generation*

Table 65. Asymmetric Encryption Conformity Testing

4.4.1.1.2.2 Digital Signature

244. For digital signature cryptographic constructions, the following conformity tests have been defined:

Digital Signature Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
RSA SHA	**PSS (PKCS#1v2.1)**	Construction (RSA-PSS) *VAL_PSS.1 for Signature Generation* *KAT_PSS.1 for Signature Verification* Underlying Primitive (RSA) *MCT_RSA.1 for modular exponentiation operations with an RSA public key* *MCT_RSA.2 for modular exponentiation operations with an RSA public key, when the value e is equal to 65537* *MCT_RSA.3 for modular exponentiation operations with an RSA private key*

		MCT_RSA.4 for modular exponentiation operations with extended parameters with a Chinese Remainder Theorem (CRT) recombination *KAT_M-R.1 for Miller-Rabin in case of RSA Key Generation* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
FFDLOG SHA	**KCDSA**	Construction (KCDSA) *VAL_KCDSA.1 for Signature Generation* *KAT_KCDSA.1 for Signature Verification* Underlying Primitive (FFDLOG) *MCT_FFDLOG.1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384*

		KAT_SHA3.3 for SHA3-512 Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
	Schnorr	Construction (Schnorr) *VAL_SDSA.1 for Signature Generation* *KAT_SDSA.1 for Signature Verification* Underlying Primitive (FFDLOG) *MCT_FFDLOG.1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
	DSA	Construction (DSA) *VAL_DSA.1 for Signature Generation* *KAT_DSA.1 for Signature Verification* Underlying Primitive (FFDLOG) *MCT_FFDLOG.1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256*

		MCT_SHA3.2 for SHA3-384 *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
ECDLOG SHA	**EC-KCDSA**	Construction (EC-KCDSA) *VAL_ECKCDSA.1 for Signature Generation* *KAT_ECKCDSA.1 for Signature Verification* Underlying Primitive (ECDLOG) *MCT_ECDLOG.1 for EC Scalar Multiplication operations* *MCT_ECDLOG.2 for EC Scalar Multiplication with Fixed Base Point operations* *MCT_ECDLOG.3 for EC Multi-scalar Multiplication operations* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
	EC-DSA	Construction (EC-DSA) *VAL_ECDSA.1 for Signature Generation* *KAT_ECDSA.1 for Signature Verification*

		Underlying Primitive (ECDLOG) *MCT_ECDLOG.1 for EC Scalar Multiplication operations* *MCT_ECDLOG.2 for EC Scalar Multiplication with Fixed Base Point operations* *MCT_ECDLOG.3 for EC Multi-scalar Multiplication operations* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
	EC-GDSA	Construction (EC-GDSA) *VAL_ECGDSA.1 for Signature Generation* *KAT_ECGDSA.1 for Signature Verification* Underlying Primitive (ECDLOG) *MCT_ECDLOG.1 for EC Scalar Multiplication operations* *MCT_ECDLOG.2 for EC Scalar Multiplication with Fixed Base Point operations* *MCT_ECDLOG.3 for EC Multi-scalar Multiplication operations* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256*

		KAT_SHA.2 for all except SHA2-256 Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384* *KAT_SHA3.3 for SHA3-512* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
	EC-Schnorr	Construction (EC-Schnorr) *VAL_ECSDSA.1 for EC-SDSA Signature Generation* *KAT_ECSDSA.1 for EC-SDSA Signature Verification* *VAL_ECFSDSA.1 for EC-FSDSA Signature Generation* *KAT_ECFSDSA.1 for EC-FSDSA Signature Verification* Underlying Primitive (ECDLOG) *MCT_ECDLOG.1 for EC Scalar Multiplication operations* *MCT_ECDLOG.2 for EC Scalar Multiplication with Fixed Base Point operations* *MCT_ECDLOG.3 for EC Multi-scalar Multiplication operations* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256* Underlying Primitive (SHA-3) *MCT_SHA3.1 for SHA3-256* *MCT_SHA3.2 for SHA3-384* *MCT_SHA3.3 for SHA3-512* *KAT_SHA3.1 for SHA3-256* *KAT_SHA3.2 for SHA3-384*

		KAT_SHA3.3 for SHA3-512 Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHA-3 family*
ECDLOG SHA2-512 SHAKE256	**EdDSA**	Construction (EdDSA) *VAL_EDDSA.1 for Signature Generation* *KAT_EDDSA.1 for Signature Verification* Underlying Primitive (ECDLOG) *MCT_ECDLOG.1 for EC Scalar Multiplication operations* *MCT_ECDLOG.2 for EC Scalar Multiplication with Fixed Base Point operations* *MCT_ECDLOG.3 for EC Multi-scalar Multiplication operations* *MCT_ECDLOG.6 for EC Base Point Multi-Scalar Multiplication operations* Underlying Primitive (SHA2-512) *MCT_SHA.1* *KAT_SHA.2* Underlying Primitive (SHAKE256) *KAT_SHAKE.2* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHAKE256*
RSA SHA	**PKCS#1v1.5**	Construction (RSA-PKCS#1v1.5) *VAL_PKCS.1 for Signature Generation* *KAT_PKCS.2 for Signature Verification* Underlying Primitive (RSA) *MCT_RSA.1 for modular exponentiation operations with an RSA public key* *MCT_RSA.2 for modular exponentiation operations with an RSA public key, when the value e is equal to 65537*

		MCT_RSA.3 for modular exponentiation operations with an RSA private key *MCT_RSA.4 for modular exponentiation operations with extended parameters with a Chinese Remainder Theorem (CRT) recombination* *KAT_M-R.1 for Miller-Rabin in case of RSA Key Generation* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256*
SHA2-256 SHAKE256	**XMSS**	Construction (XMSS) *VAL_XMSS.1 for Signature Generation* *KAT_XMSS .1 for Signature Verification* Underlying Primitive (SHA2-256) *MCT_SHA.1* *KAT_SHA.1* Underlying Primitive (SHAKE256) *KAT_SHAKE.2* Underlying Primitive (Keccak) *MCT_KECCAK.1 for SHAKE256*

Table 66. Digital Signature Conformity Testing

4.4.1.1.2.3 Key Establishment

245. For key establishment cryptographic constructions, the following conformity tests have been defined:

Key Establishment Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
FFDLOG KDF	**DH**	Construction (DH) *VAL_DH.1 in case of Key Generation* *VAL_DH.2* *KAT_DH.1*

		Underlying Construction (KDF) *See Table 63 for underlying KDF* Underlying Primitives (FFDLOG) *MCT_FFDLOG.1*
FFDLOG KDF	**DLIES-KEM**	Construction (DLIES-KEM) *KAT_DLIES-KEM.1* Underlying Construction (KDF) *See Table 63 for underlying KDF* Underlying Primitive (FFDLOG) *MCT_FFDLOG.1*
ECDLOG KDF	**EC-DH**	Construction (DH) *VAL_ECDH.1 in case of Key Generation* *VAL_ECDH.2* *KAT_ECDH.1* Underlying Construction (KDF) *See Table 63 for underlying KDF* Underlying Primitive (ECDLOG) *MCT_ECDLOG.1 for EC Scalar Multiplication operations with regular curves (NIST-P curves, Brainpool-P curves, and FRP256v1 curve)* *MCT_ECDLOG.2 for EC Scalar Multiplication with Fixed Base Point operations with regular curves (NIST-P curves, Brainpool-P curves, and FRP256v1 curve)* *MCT_ECDLOG.4 for EC Scalar Multiplication operations for Curve25519 and Curve448* *MCT_ECDLOG.5 for EC Scalar Multiplication with the Fixed Base point operations for Curve25519 and Curve448*
ECDLOG KDF	**ECIES-KEM**	Construction (ECIES-KEM) *KAT_ECIES-KEM.1* Underlying Construction (KDF) *See Table 63 for underlying KDF*

		Underlying Primitive (ECDLOG) *MCT_ECDLOG.1 for EC Scalar Multiplication operations* *MCT_ECDLOG.2 for EC Scalar Multiplication with Fixed Base Point operations*

Table 67. Key Establishment Conformity Testing

4.4.1.1.3 Deterministic Random Number Generators

246. For deterministic random number generator (DRNG) constructions, the following conformity tests have been defined:

DRNG Conformity Testing		
Underlying	**Scheme**	**Conformity Test ID**
HMAC	**HMAC-DRBG**	Construction (HMAC-DRBG) *KAT_HMAC-DRBG.1 for NR configuration* *KAT_HMAC-DRBG.2 for NPR configuration* *KAT_HMAC-DRBG.3 for PR configuration* Underlying Construction (HMAC) *KAT_HMAC.1 for all except HMAC-SHA2-512/256 and SHA-1* *KAT_HMAC.2 for SHA2-512/256* *KAT_HMAC-1.1 for SHA-1* Underlying Primitive (SHA-2) *MCT_SHA.1* *KAT_SHA.1 for SHA2-256* *KAT_SHA.2 for all except SHA2-256*
SHA	**HASH-DRBG**	Construction (Hash-DRBG) *KAT_HASH-DRBG.1 for NR configuration* *KAT_HASH-DRBG.2 for NPR configuration* *KAT_HASH-DRBG.3 for PR configuration* Underlying Primitive (SHA-2) *MCT_SHA.1 for SHA-2*

		KAT_SHA.1 for SHA2-256 *KAT_SHA.2 for all except SHA2-256*
AES-CTR	**CTR-DRBG**	Construction (CTR-DRBG) *KAT_CTR-DRBG.1 for NR with DF configuration* *KAT_CTR-DRBG.2 for NR without DF configuration* *KAT_CTR-DRBG.3 for NPR with DF configuration* *KAT_CTR-DRBG.4 for NPR without DF configuration* *KAT_CTR-DRBG.5 for PR with DF configuration* *KAT_CTR-DRBG.6 for PR without DF configuration* Underlying Construction (AES-CTR) *KAT_CTR.1* Underlying Primitive (AES) *MCT_AES.1*

Table 68. DRNG Conformity Testing

5 AVOIDANCE OF IMPLEMENTATION PITFALLS

5.1 OBJECTIVE

247. This chapter defines the common implementation pitfalls for the cryptographic mechanisms implemented by the TOE and how to avoid them. The common implementation pitfalls have been defined by CCN according to the [CCN-STIC-221] guidance in conjunction with the specified in the [SOGIS-HEP] document by the SOG-IS participants.

248. Therefore, it provides vendors with the common implementation pitfalls that shall be avoided by the TOE. In addition, it provides testers with the evaluation task necessary to verify that the TOE implements mechanisms to avoid the implementation pitfalls associated with the implemented cryptographic mechanisms.

5.2 DEFINITIONS

Cryptographic Mechanisms

249. The implementation of cryptographic mechanisms is a process where several errors might be introduced. These are common implementation errors that have been identified considering their impact on the security of the implemented mechanisms. These errors may be linked to non-conformities of the implementation or bad implementation choices of mechanisms weakening the security of the mechanisms.

5.3 CRYPTOGRAPHIC EVALUATION TASK

250. This section contains common implementation pitfalls that shall be avoided by the vendors in the case of implementing a TOE with cryptographic capabilities.

251. A cryptographic evaluation task related to the avoidance of implementation pitfalls has been defined:

CRYPTOGRAPHIC EVALUATION TASK	CCN-PITFALL	CERTIFICATION LEVEL
The **tester shall** verify the proper avoidance of the implementation pitfalls associated with all the cryptographic mechanisms implemented by the TOE (even if they are not in use).		**CL2 and CL3**

252. For the correct completion of the entire evaluation task, it has been divided into a set of cryptographic evaluation tests that the **tester shall** perform if applicable for the TOE.

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
CCN-PITFALL Avoidance of Implementation Pitfalls	CCN-PITFALL/SymEncryption.1	Impl-Dep
	CCN-PITFALL/SymEncryption.2	Impl-Dep
	CCN-PITFALL/CTR	Impl-Dep
	CCN-PITFALL/DiskEncryption	Impl-Dep
	CCN-PITFALL/XTS	Impl-Dep
	CCN-PITFALL/CBCMAC	Impl-Dep
	CCN-PITFALL/AuthEncryption	Impl-Dep
	CCN-PITFALL/AuthEncryptionComb.1	Impl-Dep
	CCN-PITFALL/AuthEncryptionComb.2	Impl-Dep
	CCN-PITFALL/CCM	Impl-Dep
	CCN-PITFALL/GCM.1	Impl-Dep
	CCN-PITFALL/GCM.2	Impl-Dep
	CCN-PITFALL/EAX	Impl-Dep
	CCN-PITFALL/SIV.1	Impl-Dep
	CCN-PITFALL/SIV.2	Impl-Dep
	CCN-PITFALL/Keywrap	Impl-Dep
	CCN-PITFALL/KeyDerivation	Impl-Dep
	CCN-PITFALL/RSA	Impl-Dep
	CCN-PITFALL/RSAParameter.1	Impl-Dep
	CCN-PITFALL/RSAParameter.2	Impl-Dep
	CCN-PITFALL/FFDLOG.1	Impl-Dep
	CCN-PITFALL/FFDLOG.2	Impl-Dep

Cryptographic Evaluation Task	Cryptographic Evaluation Test	Test Categorization
	CCN-PITFALL/ECDLOG	Impl-Dep
	CCN-PITFALL/AsymConstruction	Impl-Dep
	CCN-PITFALL/AsymEncryption	Impl-Dep
	CCN-PITFALL/RSAPKCS	Impl-Dep
	CCN-PITFALL/AsymAuth	Impl-Dep
	CCN-PITFALL/XMSS.1	Impl-Dep
	CCN-PITFALL/XMSS.2	Impl-Dep
	CCN-PITFALL/XMSS.3	Impl-Dep
	CCN-PITFALL/XMSS.4	Impl-Dep
	CCN-PITFALL/KeyEstablishment	Impl-Dep
	CCN-PITFALL/FFKeyEstablishment	Impl-Dep
	CCN-PITFALL/ECKeyEstablishment	Impl-Dep
	CCN-PITFALL/DRG.1	Impl-Dep
	CCN-PITFALL/DRG.2	Impl-Dep
	CCN-PITFALL/CTRDRBG	Impl-Dep
	CCN-PITFALL/KeyManagement	Impl-Dep
	CCN-PITFALL/KeyTransport	Impl-Dep
	CCN-PITFALL/KeyUsage	Impl-Dep

Table 69. Cryptographic Evaluation Tests defined for CCN Pitfall

5.3.1 CRYPTOGRAPHIC EVALUATION TEST DEFINITION

253. This subsection defines the cryptographic evaluation tests that shall be performed by the tester to accomplish the evaluation tasks, based on the avoidance of implementation pitfalls.

5.3.1.1 CRYPTOGRAPHIC MECHANISMS IMPLEMENTATION

254. This topic defines the cryptographic evaluation process to verify the correct avoidance of implementation pitfalls for each cryptographic mechanism implemented by the TOE, in order to fulfill the **CCN-PITFALL** evaluation task.

5.3.1.1.1 Symmetric Constructions

255. Symmetric constructions are built upon symmetric elementary primitives and enable them to address a large variety of security objectives. The security of keyed symmetric schemes relies on the confidentiality and integrity of a secret key shared by legitimate parties.

5.3.1.1.1.1 Symmetric Encryption (Confidentiality Only)

256. In case the TOE implements symmetric encryption cryptographic constructions to provide only confidentiality to the processed information:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with symmetric encryption cryptographic constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/SymEncryption.1	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that the TOE implementation ensures that the expected property of the initial vector or the counter is satisfied: - For CBC, CBC-CS, and CFB modes, the IV must be unpredictable. - For the OFB mode, the IV must only be unique for each encryption process. In this case, the IV is called nonce. - For the CTR mode, the counters shall be generated in such a way that all counters are unique across all plaintexts (and not only within a single plaintext). **Analysis:** For OFB, reusing a nonce with the same key voids the confidentiality guarantees of the mode. For the CTR mode, the standard [SP800-38A] specifies input blocks that are encrypted to produce the key stream. Using a counter twice also voids the confidentiality guarantee of the mode. Appendix B of [SP800-38A] recommends two methods to generate such counters. Also, the generation method of the first counter and the next counters for GCM [SP800-38D] in section 8.2.1 is more formal and accurate. This method can be used for the CTR mode as well. In this case, the first counter is generated from the concatenation of an IV and a counter buffer initialized with zero. For this method, the unicity of the IV guarantees the unicity of all counters.	**I5** **I6**

CCN-PITFALL/SymEncryption.1	Inputs
For CBC, CBC-CS, and CFB, reusing an IV with the same key leaks some information about the first block of the message. In addition, it must be unpredictable. [SP800-38A] recommends, in Appendix C, two methods to generate unpredictable IV. The first method is to generate an IV by encrypting a nonce with the same key that is used for the encryption of the plaintext. The second method is to generate a random IV with a strong DRNG. Overall, especially for stream modes of operation, care shall be taken when a nonce or IV is chosen at random. This is due to the birthday paradox. As an example, if a nonce or IV of size 64 bits is chosen (for example if a random nonce of size 64 bits is concatenated with a 64-bit buffer block), then the probability of collision is greater than 2^{-32} after 92681 draws. The number of draws corresponds to the number of messages being encrypted. Thus, the possible number of messages encrypted with the same key shall be analyzed to conclude this test. For 96 bits, the probability that a collision occurs is greater than 2^{-32} after 6074000999 draws, which can be an issue for very long lifetime keys. Determining whether each nonce is unique and each IV is unique and unpredictable is not achievable through known answer tests. It rather requires an analysis of the cryptographic specification's rationale of the implementation, possibly supported by a source code analysis. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	

257. In case the TOE implements symmetric encryption cryptographic constructions to provide only confidentiality to the processed information based on padding schemes:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with symmetric encryption cryptographic constructions based on padding schemes.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/SymEncryption.2	Inputs
In the case of **CL2** evaluations, verify using **I5** that an attacker has no access to the correctness of the padding or format of the deciphered ciphertext (it corresponds to the output blocks when applying the decryption mode of operation. It is equal to the "plaintext" if no padding is used. It is equal to the "padded plaintext" if padding is used in the scheme, with possibly an incorrect padding) of arbitrary ciphertexts, chosen by the attacker. **Analysis:** This applies in particular to schemes with paddings and the attack is then called a padding oracle attack [Vau02]. However, this test more generally applies to all schemes where any verification is performed on the format of the deciphered ciphertext. In this case, the attack is called a format oracle attack. Such attacks can be used to decrypt any ciphertext. It is pointed out that these attacks can be mounted if the attacker can submit arbitrary data to be deciphered. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

258. In case the TOE implements symmetric encryption cryptographic constructions to provide only confidentiality to the processed information based on CTR operation mode:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with symmetric encryption cryptographic constructions based on CTR operation mode.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/CTR	Inputs
In the case of **CL2** evaluations, verify using **I5** that the counter block is unique for each plaintext block that is encrypted under a given key. **Analysis:** Appendix B.2 of [SP800-38A] provides two methods to select the initial counter value for the counter mode to satisfy this uniqueness requirement. **CCN-PITFALL.1/SymEncryption.1** can be ignored because of this implementation pitfall. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.1.2 Disk Encryption

259. In case the TOE implements symmetric encryption cryptographic constructions to perform disk encryption operations:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with disk encryption cryptographic constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/DiskEncryption	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that the implementation ensures that the tweak value (for XTS) is unique for each encryption with the same key. **Analysis:** Indeed, like a nonce, the tweak value must be unique. This is similar to **CCN-PITFALL.1/SymEncryption.1**. The tweak value is usually derived from the address where the ciphertext is stored. Extra care shall be applied for storage devices or disk drivers that determine the used tweak from logical address rather than physical address. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

260. In case the TOE implements disk encryption cryptographic constructions based on XTS operation mode:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with disk encryption cryptographic constructions based on XTS operation mode.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/XTS	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that the implementation checks that $key_1 \neq key_2$ before using the keys in the XTS algorithm to process data with them. **Analysis:** An improper key generation that results in key1 = key2 introduces a vulnerability to chosen ciphertext attacks. Therefore, section 6.3 of [SP800-133] rev2 provides the rules for component symmetric keys to independently generate and/or establish key_1 and key_2. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.1.3 Integrity Modes: Message Authentication Code

261. In case the TOE implements message authentication code cryptographic constructions based on the CBC-MAC scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with message authentication code cryptographic constructions based on CBC-MAC.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/CBCMAC	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that the size of all the inputs for which CBC-MAC is computed under the same key is the same. **Analysis:** Otherwise, the attacker can trivially forge MAC with two known pairs (message, MAC). In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.1.4 Authenticated Encryption (AE)

262. In case the TOE implements authenticated encryption cryptographic constructions:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with authenticated encryption cryptographic constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/AuthEncryption	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that, if the integrity of the ciphertext is not properly checked before decryption, the attacker cannot access the specific error condition (format error or MAC error) that can occur in the system. **Analysis:** This applies to schemes with paddings in order to avoid so-called padding oracle attacks [Vau02]. However, the task applies to all schemes if any verification is performed on the format of the deciphered ciphertext and if the deciphered ciphertexts are sent to consuming applications. Such verifications may lead to a format oracle attack. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

263. In case the TOE implements authenticated encryption cryptographic constructions based on the Encrypt-then-MAC scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with authenticated encryption cryptographic constructions based on the Encrypt-then-MAC scheme.
- The **tester shall** perform the following cryptographic evaluation tests:

CCN-PITFALL/AuthEncryptionComb.1	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that the implementation pitfalls of the underlying mechanisms are considered. In particular, the ones referring to symmetric encryption and MAC constructions. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/AuthEncryptionComb.2	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that the keys used for encryption and MAC operations are different. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

264. In case the TOE implements authenticated encryption cryptographic constructions based on the CCM scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with authenticated encryption cryptographic constructions based on the CCM scheme.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/CCM	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that all the implementation pitfalls of the CTR operation mode and the CBC-MAC scheme are considered. **Analysis:** CCM relies on the CTR mode for encryption and the CBC-MAC mechanism for authentication. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

265. In case the TOE implements authenticated encryption, cryptographic constructions based on the GCM scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with authenticated encryption cryptographic constructions based on the GCM scheme.
- The **tester shall** perform the following cryptographic evaluation tests:

CCN-PITFALL/GCM.1	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that no message of length strictly greater than 2^{32} - 2 blocks can be encrypted. **Analysis:** The counters are generated with the concatenation of a unique IV of 96 bits and an incremented counter of 32 bits. This test avoids the overflow of the counter. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/GCM.2	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that an IV can never be reused in encryption under the same key with differing inputs. **Analysis:** GCM relies on the CTR mode. Then, all the implementation pitfalls of the CTR operation shall be considered. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

266. In case the TOE implements authenticated encryption cryptographic constructions based on the EAX scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with authenticated encryption cryptographic constructions based on the EAX scheme.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/EAX	Inputs
In the case of **CL2** evaluations, verify using **I5** that all the implementation pitfalls of the CTR operation mode and the CMAC scheme are considered. **Analysis:** EAX relies on the CTR mode for encryption and the OMAC mechanism (which is very similar or equivalent to CMAC) for authentication. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.1.5 Key Protection

267. In case the TOE implements key protection cryptographic constructions based on the SIV scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key protection cryptographic constructions based on the SIV scheme.
- The **tester shall** perform the following cryptographic evaluation tests:

CCN-PITFALL/SIV.1	Inputs
In the case of **CL2** evaluations, verify using **I5** that all the implementation pitfalls of the CTR operation mode and the CMAC scheme are considered. **Analysis:** SIV relies on the CTR mode for encryption and the CMAC mechanism for authentication. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/SIV.2	Inputs
In the case of **CL2** evaluations, verify using **I5** that, when the SIV scheme uses associated data, at most n - 2 associated data components are used as inputs of the SIV scheme, where n denotes the block size of the underlying block cipher. **Analysis:** As AES is the only underlying function in SIV specification [RFC-5297], it limits the number of associated data to 126. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

268. In case the TOE implements key protection cryptographic constructions based on the Keywrap scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key protection cryptographic constructions based on the Keywrap scheme.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/Keywrap	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that plaintexts are not larger than 2^{54} - 1 semi blocks for Key Wrap without padding and 2^{32} - 1 bytes for Key Wrap with padding [SP800-38F]. **Analysis:** This is to avoid any collision. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

Note: In case the TOE implements key protection based on Authenticated Encryption cryptographic mechanisms, the **tester shall** perform the evaluation tests from section 5.3.1.1.1.4 "Authenticated Encryption (AE)" depending on the implemented mechanisms.

5.3.1.1.1.6 Key Derivation Functions (KDF)

269. In case the TOE implements key derivation cryptographic constructions:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key derivation cryptographic constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/KeyDerivation	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that invalid requests for the keying data generation are not possible. **Analysis:** The computation of the derived key starts with some size controls and that shall not be ignored. In particular, the tester shall verify that no derived key is larger than: - $255 \times h$ for HKDF constructions - $(2^{32} - 1) \times h$ for the rest of the key derivation functions where h is the length (in bits) of the output block of the underlying hash function or pseudo-random function. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.2 Asymmetric Elementary Primitives

5.3.1.1.2.1 RSA/Integer Factorization

270. In case the TOE implements asymmetric cryptographic primitives based on the integer factorization problem (RSA):

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with an asymmetric cryptographic primitive based on the integer factorization problem (RSA).
- The **tester shall** perform the following cryptographic evaluation tests:

CCN-PITFALL/RSA	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that all RSA public exponents in the system are greater than 2^{16}. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned. **Note:** If the TOE implements RSA key pair generation, the successful verification of the **CCN-AGREED/RSAKeys** evaluation test implies the successful verification of this test. The inverse is not true.	**I5** **I6**

CCN-PITFALL/RSAParameter.1	**Inputs**
In the case of **CL2** evaluations, verify using **I5** that p and q, have the same bit length and that their product has the required modulus bit length. Moreover, the following condition shall be satisfied: $$\lvert p - q \rvert \geq 2^{\frac{n}{2}-100}$$ where *n* denotes the bit length of the modulus. **Analysis:** For this test, the tester shall check if the condition is implemented in the source code or check the randomness of p and q. In the latter case, the probability that the condition is not satisfied is negligible [FIPS-186-4]. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned. **Note:** If the TOE implements RSA key pair generation, the successful verification of the **CCN-AGREED/RSAKeys** evaluation test implies the successful verification of this test. The inverse is not true.	**I5** **I6**

CCN-PITFALL/RSAParameter.2	Inputs
In the case of **CL2** evaluations, verify using **I5** that d is larger than $2^{n/2}$, where n denotes the bit length of the modulus. **Analysis:** For this test, the tester shall check if the condition is implemented in the source code or check that d has been chosen after a small public exponent e has been generated. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned. **Note:** If the TOE implements RSA key pair generation, the successful verification of the **CCN-AGREED/RSAKeys** evaluation test implies the successful verification of this test. The inverse is not true.	**I5** **I6**

5.3.1.1.2.2 Multiplicative Discrete Logarithm Problem

271. In case the TOE implements asymmetric cryptographic primitives based on FFDLOG:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with asymmetric cryptographic primitives based on FFDLOG.
- The **tester shall** perform the following cryptographic evaluation tests:

CCN-PITFALL/FFDLOG.1	Inputs
In the case of **CL2** evaluations, verify using **I5** that r is greater than 200 bits. **Analysis:** In addition, it is recommended that the size of r be greater than 250 bits. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/FFDLOG.2	Inputs
In the case of **CL2** evaluations, verify using **I5** that the system only manipulates correct values y. These values must have order divisible by r and dividing q. Particularly, points received from another entity shall be verified by the system. **Analysis:** This requirement ensures that the manipulated values do not lie in subgroups of small size (i.e., ≤ 250 bits) or outside the intended group.	**I5** **I6**

CCN-PITFALL/FFDLOG.2	Inputs
If q is prime, $r = q$ and the system shall verify that manipulated values have exact order q. It is advised to pick a subgroup of prime numbers. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	

5.3.1.1.2.3 Additive Discrete Logarithm Problem

272. In case the TOE implements asymmetric cryptographic primitives based on ECDLOG:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with asymmetric cryptographic primitives based on ECDLOG.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/ECDLOG	Inputs
In the case of **CL2** evaluations, verify using **I5** that the system only manipulates points lying on the intended curve. In addition, if a curve with a non-prime order is used, the points shall be in the intended subgroup. Particularly, points received from another entity shall be verified. **Analysis:** This test is required when points are received from another entity. In this case, the system shall verify that, before being used, the manipulated points lie on the curve and are in the intended subgroup. Otherwise, the Invalid Curve Attack [JSS15] can be possible. If q is prime, $r = q$ and the tester shall verify that manipulated points have exact order q. It is advised to pick a subgroup of prime numbers. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.3 Asymmetric Constructions

273. In case the TOE implements asymmetric cryptographic constructions:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with asymmetric cryptographic constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/AsymConstruction	Inputs
In the case of **CL2** evaluations, verify using **I5** that all the implementation pitfalls of the underlying mechanisms are considered. **Analysis:** Pitfalls related to the asymmetric primitives, the hash functions, the DRG, and/or block ciphers shall be considered. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.3.1 Asymmetric Encryption Scheme

274. In case the TOE implements asymmetric encryption cryptographic constructions based on the RSA asymmetric primitive:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with asymmetric encryption cryptographic constructions based on the RSA primitive.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/AsymEncryption	Inputs
In the case of **CL2** evaluations, verify using **I5** that the attacker cannot access the specific error condition that can occur during the decoding of an encoded message. **Analysis:** If the attacker can distinguish errors, attacks such as the chosen ciphertext attack [Man01] for OAEP can be performed. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

275. In case the TOE implements asymmetric encryption cryptographic constructions based on the RSA-PKCS#1v1.5 scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with asymmetric encryption cryptographic constructions based on the RSA-PKCS#1v1.5 scheme.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/RSAPKCS	Inputs
In the case of **CL2** evaluations, verify using **I5** that a padding oracle is not available to an attacker, or that all attacks similar to Bleichenbacher's attack are taken care of with efficient countermeasures. **Analysis:** Bleichenbacher's attack is an example of an attack targeting this encryption scheme using an oracle. In the case where countermeasures are applied, the tester shall equivalently verify that the system handles in the same way: - An incorrectly encoded ciphertext, and, - A correctly encoded ciphertext that has been obtained from a relation with another valid ciphertext rather than the encryption operation. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.3.2 Digital Signature

276. In case the TOE implements digital signature cryptographic constructions:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with digital signature cryptographic constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/AsymAuth	Inputs
In the case of **CL2** evaluations, verify using **I5** that the private key is not used for different purposes (signature and authentication). In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

277. In case the TOE implements digital signature constructions based on the XMSS scheme:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with digital signature constructions based on the XMSS scheme.
- The **tester shall** perform the following cryptographic evaluation tests:

CCN-PITFALL/XMSS.1	Inputs
In the case of **CL2** evaluations, verify using **I5** that the private key is updated each time a message is signed. **Analysis:** The method employed by the TOE to perform the verification shall be evaluated. If the key is stored in non-volatile memory, it must be ensured that the private key is updated in the non-volatile memory before exporting the corresponding signature. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/XMSS.2	Inputs
In the case of **CL2** evaluations, verify using **I5** that the same private key and the same counter are not used to sign different documents. **Analysis:** The method employed by the TOE to perform the verification shall be evaluated. Moreover, appropriate precautions must be taken to prevent this issue during the process of recovering a backup copy. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/XMSS.3	Inputs
In the case of **CL2** evaluations, verify using **I5** that, before generating the signature of a message, the private key counter does not exceed the maximum allowed by the established parameters. **Analysis:** The method employed by the TOE to perform the verification shall be evaluated. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/XMSS.4	Inputs
In the case of **CL2** evaluations, verify using **I5** that the derivation of the *WOTS+* keys is performed as specified in [SP800-208], that is, using the *PRF_keygen* function to prevent multi-target attacks. **Analysis:** The method employed by the TOE to perform the verification shall be evaluated. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.3.3 Key Establishment

278. In case the TOE implements key establishment cryptographic constructions:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key establishment cryptographic constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/KeyEstablishment	Inputs
In the case of **CL2** evaluations, verify using **I5** that, if a secret is generated from a preexisting secret, this preexisting secret has at least 100 bits of entropy. **Analysis:** Ephemeral keys generated using the Diffie-Hellman scheme and master secret used in TLS record protocol are examples of such preexisting secrets. In accordance with the entropy of such preexisting secrets, e.g., each ephemeral Diffie-Hellman private key in the former example, must be at least 100 bits for legacy mechanisms and 125 bits for recommended mechanisms. The successful verification of the **CCN-AGREED/TRNGEntropy** evaluation test ensures that the average entropy per internal random bit exceeds a certain minimum close to 1. If the previous minimum is met, verifying that a bit string contains at least 100 bits of entropy entails checking that its length is at least 101 bits. Verifying that a bit string contains at least 125 bits of entropy entails checking that its length is at least 126 bits. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

279. In case the TOE implements key establishment cryptographic constructions based on FFDLOG primitive:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key establishment cryptographic constructions based on FFDLOG.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/FFKeyEstablishment	Inputs
In the case of **CL2** evaluations, verify using **I5** that **CCN-PITFALL/FFDLOG.1** and **CCN-PITFALL/FFDLOG.2** are considered. **Analysis:** The manipulated values have order divisible by r and dividing q. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

280. In case the TOE implements key establishment cryptographic constructions based on ECDLOG primitive:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key establishment cryptographic constructions based on ECDLOG.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/ECKeyEstablishment	Inputs
In the case of **CL2** evaluations, verify using **I5** that **CCN-PITFALL/ECDLOG** is considered. **Analysis:** The point from the other entity shall lie on the curve and be in the intended subgroup. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.4 Random Number Generators

5.3.1.1.4.1 Deterministic Random Number Generators

281. In case the TOE implements deterministic random number generators (DRNG):

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with deterministic random number generator (DRNG) implementations.
- The **tester shall** perform the following cryptographic evaluation tests:

CCN-PITFALL/DRG.1	Inputs
In the case of **CL2** evaluations, verify using **I5** that the entropy of the seed is at least 125 bits long. For this purpose, identify the random sources used to seed for instantiating and reseed for updating the internal state of the DRG, and analyze their random quality. **Analysis:** The successful verification of the **CCN-AGREED/TRNGEntropy** evaluation test ensures that the average entropy per internal random bit exceeds a certain minimum close to 1. If the previous minimum is met, verifying that the seed contains at least 125 bits of entropy entails checking that its length is at least 126 bits. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

CCN-PITFALL/DRG.2	Inputs
In the case of **CL2** evaluations, verify using **I5** that the internal state of the DRG is periodically reseeded with the random source to ensure enough entropy in case of the generation of a large random quantity. **Analysis:** In some implementations (such as smartcards), reseeding may not be possible and an alternative to reseeding shall be to create an entirely new instantiation (to obtain a new seed and guarantee enough entropy in the DRG internal state). In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

282. In case the TOE implements deterministic random number generators (DRNG) based on CTR symmetric encryption constructions:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with deterministic random number generator (DRNG) based on CTR symmetric encryption constructions.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/CTRDRBG	Inputs
In the case of **CL2** evaluations, verify using **I5** that a single invocation of the CTR-DRBG generate function is not done for the generation of several different keys, if the property of perfect forward secrecy is required in the system. **Analysis:** The backtracking resistance of CTR-DRBG is only given for the transition function between different invocations. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.5 Key Management

283. In case the TOE implements key management procedures:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key management procedures.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/KeyManagement	Inputs
In the case of **CL2** evaluations, verify using **I5** that the variable memory allocated for a key is locked when it is manipulated by the system. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.5.1 Key Storage and Transport

284. In case the TOE implements key transport procedures:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key transport procedures.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/KeyTransport	Inputs
In the case of **CL2** evaluations, verify using **I5** that the key distribution channel is protected in authenticity and integrity for public keys and secret keys. In addition, secret keys shall be protected in confidentiality. **Analysis:** Some cryptographic keys must be distributed between identified users, allowing them to access sensitive data protected by the related cryptosystems. So, the protection of the key distribution channel is crucial. Also, if the system allows users to load their secret key generated outside the system, the tester shall check that the secret key has not been modified during import. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

5.3.1.1.5.2 Key Usage

285. In case the TOE implements key usage procedures:

- The **TOE shall** implement mechanisms to avoid implementation pitfalls associated with key usage procedures.
- The **tester shall** perform the following cryptographic evaluation test:

CCN-PITFALL/KeyUsage	Inputs
In the case of **CL2** evaluations, verify using **I5** that each key has only one usage. **Analysis:** A key must not be used with different mechanisms or contexts, otherwise an attacker could exploit a source of errors in the usage of the key. This does not forbid to derive various keys from a master key. In the case of **CL3** evaluations, use **I5** and perform a source code review (using **I6**) to verify the above-mentioned.	**I5** **I6**

6 ANNEX A: TRACEABILITY OF EVALUATION TASKS

286. This annex summarizes the evaluation tasks related to each cryptographic mechanism. The objective is to provide the tester with the complete list of evaluation tasks to be performed in order to evaluate each agreed cryptographic mechanism.

287. The reason to list the evaluation tasks for each construction instead of for each primitive is that the cryptographic functionality of the TOE relies on the execution of cryptographic constructions instead of primitives. Moreover, for each construction, the evaluation tasks related to its underlying primitives will be performed.

6.1 SYMMETRIC CRYPTOGRAPHIC MECHANISMS

6.1.1 HASH FUNCTIONS

288. The following table summarizes the evaluation tasks to be performed for the **SHA-2** family of the hash cryptographic functions:

Summary of Evaluation Tasks for SHA-2 Primitives	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Primitive** CCN-MECHANISM/Hash
CCN-AGREED	**Primitive** CCN-AGREED/Hash
CCN-CONFORMITY	**Primitive** CCN-CONFORMITY/Mechanisms (Table 55)
CCN-PITFALL	**Primitive** N/A because there are no implementation pitfalls for the SHA cryptographic primitive

Table 70. Summary of Evaluation Tasks for SHA-2 Primitives

289. The following table summarizes the evaluation tasks to be performed for the **SHA-3** family of the hash cryptographic functions:

Summary of Evaluation Tasks for SHA-3 Primitives	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Primitive** CCN-MECHANISM/Hash
CCN-AGREED	**Primitive** CCN-AGREED/Hash

Summary of Evaluation Tasks for SHA-3 Primitives	
Evaluation Task	**Evaluation Test Code**
CCN-CONFORMITY	**Primitive** CCN-CONFORMITY/Mechanisms (Table 55)
CCN-PITFALL	**Primitive** N/A because there are no implementation pitfalls for the SHA cryptographic primitive

Table 71. Summary of Evaluation Tasks for SHA-3 Primitives

6.1.2 SECRET SHARING

290. The following table summarizes the evaluation tasks to be performed for the **Secret Sharing** cryptographic primitives:

Summary of Evaluation Tasks for Secret Sharing Primitives	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Primitive** CCN-MECHANISM/SecretSharing
CCN-AGREED	**Primitive** CCN-AGREED/SecretSharing
CCN-CONFORMITY	**Primitive** N/A because there are no conformity tests for the Secret Sharing cryptographic primitive
CCN-PITFALL	**Primitive** N/A because there are no implementation pitfalls for the Secret Sharing cryptographic primitive

Table 72. Summary of Evaluation Tasks for Secret Sharing Primitives

6.1.3 SYMMETRIC ENCRYPTION (CONFIDENTIALITY ONLY)

291. The following table summarizes the evaluation tasks to be performed for the **CBC** operation mode of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for CBC Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/SymEncryption

Summary of Evaluation Tasks for CBC Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/SymEncryption **Notes** Note 4 [IV Type] Note 5 [Add Integrity] Note 7 [Padding]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 57)
CCN-PITFALL	**Construction** CCN-PITFALL/SymEncryption.1 CCN-PITFALL/SymEncryption.2
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 73. Summary of Evaluation Tasks for CBC Encryption Constructions

292. The following table summarizes the evaluation tasks to be performed for the **CBC-CS** operation mode of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for CBC-CS Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/SymEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/SymEncryption **Notes** Note 4 [IV Type] Note 5 [Add Integrity]

Summary of Evaluation Tasks for CBC-CS Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 57)
CCN-PITFALL	**Construction** CCN-PITFALL/SymEncryption.1
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 74. Summary of Evaluation Tasks for CBC-CS Encryption Constructions

293. The following table summarizes the evaluation tasks to be performed for the **CFB** operation mode of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for CFB Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/SymEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/SymEncryption **Notes** Note 4 [IV Type] Note 5 [Add Integrity] Note 7 [Padding]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 57)
CCN-PITFALL	**Construction** CCN-PITFALL/SymEncryption.1
	Underlying Primitive

Summary of Evaluation Tasks for CFB Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
	N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 75. Summary of Evaluation Tasks for CFB Encryption Constructions

294. The following table summarizes the evaluation tasks to be performed for the **CTR** operation mode of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for CTR Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/SymEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/SymEncryption **Notes** Note 4 [IV Type] Note 5 [Add Integrity] Note 6 [Stream Mode]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 57)
CCN-PITFALL	**Construction** CCN-PITFALL/SymEncryption.1 CCN-PITFALL/CTR
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 76. Summary of Evaluation Tasks for CTR Encryption Constructions

295. The following table summarizes the evaluation tasks to be performed for the **OFB** operation mode of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for OFB Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/SymEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/SymEncryption **Notes** Note 4 [IV Type] Note 5 [Add Integrity] Note 6 [Stream Mode]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 57)
CCN-PITFALL	**Construction** CCN-PITFALL/SymEncryption.1
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 77. Summary of Evaluation Tasks for OFB Encryption Constructions

6.1.4 DISK ENCRYPTION

296. The following table summarizes the evaluation tasks to be performed for the **XTS** operation mode of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for XTS Disk Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DiskEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/DiskEncryption

Summary of Evaluation Tasks for XTS Disk Encryption Constructions	
Evaluation Task	**Evaluation Test Code**
	Notes Note 8 [Unique Tweak] Note 9 [XTS Key]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 58)
CCN-PITFALL	**Construction** CCN-PITFALL/DiskEncryption CCN-PITFALL/XTS
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 78. Summary of Evaluation Tasks for XTS Disk Encryption Constructions

6.1.5 INTEGRITY MODES: MESSAGE AUTHENTICATION CODE

297. The following table summarizes the evaluation tasks to be performed for the **CMAC** scheme based on the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for CMAC Constructions based on AES	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/MAC
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/MAC **Notes** Note 10 [MAC Truncation 96 bits] Note 11 [MAC Truncation 64 bits]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 59)

Summary of Evaluation Tasks for CMAC Constructions based on AES	
Evaluation Task	**Evaluation Test Code**
CCN-PITFALL	**Construction** N/A because there are no implementation pitfalls for the CMAC cryptographic construction
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 79. Summary of Evaluation Tasks for CMAC Constructions based on AES

298. The following table summarizes the evaluation tasks to be performed for the **CBC-MAC** scheme based on the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for CBC-MAC Constructions based on AES	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/MAC
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/MAC **Notes** Note 10 [MAC Truncation 96 bits] Note 11 [MAC Truncation 64 bits] Note 12 [Fixed Input Length]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 59)
CCN-PITFALL	**Construction** CCN-PITFALL/CBCMAC
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 80. Summary of Evaluation Tasks for CBC-MAC Constructions based on AES

299. The following table summarizes the evaluation tasks to be performed for the **HMAC** scheme based on **SHA** cryptographic primitive:

Summary of Evaluation Tasks for HMAC Constructions based on SHA	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/MAC
	Underlying Primitive CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/MAC **Notes** Note 10 [MAC Truncation 96 bits] Note 11 [MAC Truncation 64 bits] Note 13 [HMAC-SHA-1]
	Underlying Primitive CCN-AGREED/Hash
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 59)
CCN-PITFALL	**Construction** N/A because there are no implementation pitfalls for the HMAC cryptographic constructions
	Underlying Primitive N/A because there are no implementation pitfalls for the SHA cryptographic primitive

Table 81. Summary of Evaluation Tasks for HMAC Constructions based on SHA

300. The following table summarizes the evaluation tasks to be performed for the **GMAC** scheme based on the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for GMAC Constructions based on AES	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/MAC
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/MAC **Notes**

Summary of Evaluation Tasks for GMAC Constructions based on AES	
Evaluation Task	**Evaluation Test Code**
	Note 14 [GMAC-GCM Nonce] Note 15 [GMAC-GCM Options] Note 16 [GMAC-GCM Bounds]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 59)
CCN-PITFALL	**Construction** N/A because there are no implementation pitfalls for the GMAC cryptographic construction
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 82. Summary of Evaluation Tasks for GMAC Constructions based on AES

301. The following table summarizes the evaluation tasks to be performed for the **KMAC** scheme based on **XOF** cryptographic primitive:

Summary of Evaluation Tasks for KMAC Constructions based on XOF	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/MAC
	Underlying Primitive CCN-MECHANISM/XOF
CCN-AGREED	**Construction** CCN-AGREED/MAC **Notes** Note 17 [KMAC Security Settings]
	Underlying Primitive CCN-AGREED/XOF **Notes** Note 3 [Agreed XOF]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 59)

Summary of Evaluation Tasks for KMAC Constructions based on XOF	
Evaluation Task	**Evaluation Test Code**
CCN-PITFALL	**Construction** N/A because there are no implementation pitfalls for the KMAC cryptographic construction
	Underlying Primitive N/A because there are no implementation pitfalls for the XOF cryptographic primitive

Table 83. Summary of Evaluation Tasks for KMAC Constructions based on XOF

6.1.6 SYMMETRIC ENTITY AUTHENTICATION SCHEMES

302. The following table summarizes the evaluation tasks to be performed for the symmetric **entity authentication** schemes (challenge-response protocol). It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in section 6.1.3 "Symmetric Encryption (Confidentiality Only)" for the underlying symmetric encryption constructions.
- The **tester shall** perform the evaluation tasks defined in section 6.1.5 "Integrity Modes: Message Authentication Code" for the underlying MAC constructions.

Summary of Evaluation Tasks for Entity Authentication Schemes	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/EntityAuth
CCN-AGREED	**Construction** CCN-AGREED/EntityAuth **Notes** Note 19 [Challenge-Response Protocol]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 60)
CCN-PITFALL	**Construction** N/A because there are no implementation pitfalls for the symmetric entity authentication schemes

Table 84. Summary of Evaluation Tasks for Entity Authentication Schemes

6.1.7 AUTHENTICATED ENCRYPTION (AE)

303. The following table summarizes the evaluation tasks to be performed for the **Encrypt-then-MAC** schemes. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in section 6.1.3 "Symmetric Encryption (Confidentiality Only)" for the underlying symmetric encryption constructions.
- The **tester shall** perform the evaluation tasks defined in section 6.1.5 "Integrity Modes: Message Authentication Code" for the underlying MAC constructions.

Summary of Evaluation Tasks for Encrypt-then-MAC Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/AuthEncryption
CCN-AGREED	**Construction** CCN-AGREED/AuthEncryption **Notes** Note 20 [Authenticated Encryption Schemes] Note 21 [Decryption Order]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 61)
CCN-PITFALL	**Construction** CCN-PITFALL/AuthEncryption CCN-PITFALL/AuthEncryptionComb.1 CCN-PITFALL/AuthEncryptionComb.2

Table 85. Summary of Evaluation Tasks Encrypt-then-MAC Constructions

304. The following table summarizes the evaluation tasks to be performed for the **CCM** scheme of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for CCM AE Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/AuthEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction**

Summary of Evaluation Tasks for CCM AE Constructions	
Evaluation Task	**Evaluation Test Code**
	CCN-AGREED/AuthEncryption **Notes** Note 20 [Authenticated Encryption Schemes] Note 21 [Decryption Order]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 61)
CCN-PITFALL	**Construction** CCN-PITFALL/AuthEncryption CCN-PITFALL/CCM
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 86. Summary of Evaluation Tasks for CCM AE Constructions

305. The following table summarizes the evaluation tasks to be performed for the **GCM** scheme of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for GCM AE Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/AuthEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/AuthEncryption **Notes** Note 14 [GMAC-GCM Nonce] Note 15 [GMAC-GCM Options] Note 16 [GMAC-GCM Bounds] Note 20 [Authenticated Encryption Schemes] Note 21 [Decryption Order] Note 22 [GCM Plaintext Length]
	Underlying Primitive

Summary of Evaluation Tasks for GCM AE Constructions	
Evaluation Task	**Evaluation Test Code**
	CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 61)
CCN-PITFALL	**Construction** CCN-PITFALL/AuthEncryption CCN-PITFALL/GCM.1 CCN-PITFALL/GCM.2
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 87. Summary of Evaluation Tasks for GCM AE Constructions

306. The following table summarizes the evaluation tasks to be performed for the **EAX** scheme of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for EAX AE Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/AuthEncryption
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/AuthEncryption **Notes** Note 20 [Authenticated Encryption Schemes] Note 21 [Decryption Order]
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 61)
CCN-PITFALL	**Construction** CCN-PITFALL/AuthEncryption CCN-PITFALL/EAX
	Underlying Primitive

Summary of Evaluation Tasks for EAX AE Constructions	
Evaluation Task	**Evaluation Test Code**
	N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 88. Summary of Evaluation Tasks for EAX AE Constructions

307. The following table summarizes the evaluation tasks to be performed for the **ChaCha20-Poly1305** symmetric cryptographic construction:

Summary of Evaluation Tasks for ChaCha20-Poly1305 AE Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/AuthEncryption
	Underlying Primitive and Construction CCN-MECHANISM/StreamCipher CCN-MECHANISM/MAC
CCN-AGREED	**Construction** CCN-AGREED/AuthEncryption **Notes** Note 23 [ChaCha20-Poly1305 IV]
	Underlying Primitive and Construction CCN-AGREED/StreamCipher CCN-AGREED/MAC **Notes** Note 1 [ChaCha20 Implementation] Note 18 [Poly1305 Implementation]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 61)
CCN-PITFALL	**Construction** CCN-PITFALL/AuthEncryption
	Underlying Primitive and Construction N/A because there are no implementation pitfalls for the ChaCha20 and Poly1305 cryptographic mechanisms

Table 89. Summary of Evaluation Tasks for ChaCha20-Poly1305 AE Constructions

6.1.8 KEY PROTECTION

308. The following table summarizes the evaluation tasks to be performed for the **SIV** scheme of the **AES** symmetric cryptographic primitive. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The tester shall perform the evaluation tasks defined in Table 76 for the CTR underlying construction.
- The tester shall perform the evaluation tasks defined in Table 79 for the CMAC underlying construction.

Summary of Evaluation Tasks for SIV Key Protection Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyProtection
CCN-AGREED	**Construction** CCN-AGREED/KeyProtection
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 62)
CCN-PITFALL	**Construction** CCN-PITFALL/SIV.1 CCN-PITFALL/SIV.2

Table 90. Summary of Evaluation Tasks for SIV Key Protection Constructions

309. The following table summarizes the evaluation tasks to be performed for the **Keywrap** scheme of the **AES** symmetric cryptographic primitive:

Summary of Evaluation Tasks for Keywrap Key Protection Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyProtection
	Underlying Primitive CCN-MECHANISM/BlockCipher
CCN-AGREED	**Construction** CCN-AGREED/KeyProtection
	Underlying Primitive CCN-AGREED/BlockCipher
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 62)

Summary of Evaluation Tasks for Keywrap Key Protection Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-PITFALL	**Construction** CCN-PITFALL/Keywrap
	Underlying Primitive N/A because there are no implementation pitfalls for the AES cryptographic primitive

Table 91. Summary of Evaluation Tasks for Keywrap Key Protection Constructions

Note: In case the TOE implements key protection based on Authenticated Encryption cryptographic mechanisms, the **tester shall** perform the evaluation task detailed in section 6.1.7 "Authenticated Encryption (AE)" depending on the implemented mechanisms.

6.1.9 KEY DERIVATION FUNCTIONS

310. The following table summarizes the evaluation tasks to be performed for the **NIST SP800-56 A/B/C** key derivation cryptographic construction. It is mandatory to perform the evaluation tasks associated with the underlying mechanisms.

- For NIST SP800-56 A/B/C OneStep KDF based in HMAC:
  - The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC underlying constructions.
- For NIST SP800-56 A/B/C OneStep KDF based in KMAC:
  - The **tester shall** perform the evaluation tasks defined in Table 83 for the KMAC underlying constructions.
- For NIST SP800-56 A/B/C OneStep KDF based in SHA:
  - The **tester shall** perform the evaluation tasks defined in Table 70 and/or Table 71 for the hash underlying primitive.
- For NIST SP800-56 A/B/C TwoStep KDF based in HMAC:
  - The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC underlying constructions.
  - The **tester shall** perform the evaluation tasks defined in Table 93 for the SP800-108 KDF underlying constructions based on HMAC.
- For NIST SP800-56 A/B/C TwoStep KDF based in CMAC:
  - The **tester shall** perform the evaluation tasks defined in Table 79 for the CMAC underlying constructions.
  - The **tester shall** perform the evaluation tasks defined in Table 93 for the SP800-108 KDF underlying constructions based on CMAC.

Summary of Evaluation Tasks for NIST SP800-56 A/B/C KDF	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyDerivation
CCN-AGREED	**Construction** CCN-AGREED/KeyDerivation
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 63)
CCN-PITFALL	**Construction** CCN-PITFALL/KeyDerivation

Table 92. Summary of Evaluation Tasks for NIST SP800-56 A/B/C KDF

311. The following table summarizes the evaluation tasks to be performed for the **NIST SP800-108** key derivation cryptographic construction. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in Table 79 for the CMAC underlying constructions.
- The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC underlying constructions.
- The **tester shall** perform the evaluation tasks defined in Table 83 for the KMAC underlying constructions.

Summary of Evaluation Tasks for NIST SP800-108 KDF	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyDerivation
CCN-AGREED	**Construction** CCN-AGREED/KeyDerivation
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 63)
CCN-PITFALL	**Construction** CCN-PITFALL/KeyDerivation

Table 93. Summary of Evaluation Tasks for NIST SP800-108 KDF

312. The following table summarizes the evaluation tasks to be performed for the **ANSI-X9.63** key derivation cryptographic construction:

Summary of Evaluation Tasks for ANSI-X9.63 KDF	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyDerivation
	Underlying Primitive CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/KeyDerivation
	Underlying Primitive CCN-AGREED/Hash
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 63)
CCN-PITFALL	**Construction** CCN-PITFALL/KeyDerivation
	Underlying Primitive N/A because there are no implementation pitfalls for the SHA cryptographic primitive

Table 94. Summary of Evaluation Tasks for ANSI-X9.63 KDF

313. The following table summarizes the evaluation tasks to be performed for the **PBKDF2** key derivation cryptographic construction. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC underlying constructions.

Summary of Evaluation Tasks for PBKDF2 Key Derivation Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyDerivation
CCN-AGREED	**Construction** CCN-AGREED/KeyDerivation **Notes** Note 24 [PBKDF2 PRF]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 63)
CCN-PITFALL	**Construction**

Summary of Evaluation Tasks for PBKDF2 Key Derivation Constructions	
Evaluation Task	**Evaluation Test Code**
	CCN-PITFALL/KeyDerivation

Table 95. Summary of Evaluation Tasks for PBKDF2 Key Derivation Constructions

314. The following table summarizes the evaluation tasks to be performed for the **HKDF** key derivation cryptographic construction. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC underlying constructions.

Summary of Evaluation Tasks for HKDF Key Derivation Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyDerivation
CCN-AGREED	**Construction** CCN-AGREED/KeyDerivation
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 63)
CCN-PITFALL	**Construction** CCN-PITFALL/KeyDerivation

Table 96. Summary of Evaluation Tasks for HKDF Key Derivation Constructions

6.1.10 PASSWORD PROTECTION/HASHING MECHANISMS

315. The following table summarizes the evaluation tasks to be performed for the **Argon2id** password protection/hashing mechanism.

Summary of Evaluation Tasks for Argon2id Password Protection	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/PasswordMechanisms
	Underlying Primitive CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/PasswordMechanisms **Notes**

Summary of Evaluation Tasks for Argon2id Password Protection	
Evaluation Task	**Evaluation Test Code**
	Note 25 [Use of Argon2id] Note 26 [Argon2id Parameters]
	Underlying Primitive CCN-AGREED/Hash **Notes** Note 2 [Agreed BLAKE2b]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 64)
CCN-PITFALL	**Construction** N/A because there are no implementation pitfalls for the Argon2id cryptographic construction
	Underlying Primitive N/A because there are no implementation pitfalls for the BLAKE2b cryptographic primitive

Table 97. Summary of Evaluation Tasks for Argon2id Password Protection

316. The following table summarizes the evaluation tasks to be performed for the **PBKDF2** password protection/hashing mechanism. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC underlying constructions.

Summary of Evaluation Tasks for PBKDF2 Password Protection	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/PasswordMechanisms
CCN-AGREED	**Construction** CCN-AGREED/PasswordMechanisms **Notes** Note 27 [Number of Iterations] Note 28 [Salt]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 64)
CCN-PITFALL	**Construction** CCN-PITFALL/KeyDerivation

Table 98. Summary of Evaluation Tasks for PBKDF2 Password Protection

317. The following table summarizes the evaluation tasks to be performed for the **SCRYPT** password protection/hashing mechanism. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in Table 98 for the PBKDF2 underlying construction.
- The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC-SHA2-256 underlying construction.

Summary of Evaluation Tasks for SCRYPT Key Derivation Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyDerivation
CCN-AGREED	**Construction** CCN-AGREED/KeyDerivation
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 64)
CCN-PITFALL	**Construction** CCN-PITFALL/KeyDerivation

Table 99. Summary of Evaluation Tasks for SCRYPT Password Protection

6.2 ASYMMETRIC CRYPTOGRAPHIC MECHANISMS

6.2.1 ASYMMETRIC ENCRYPTION SCHEMES

318. The following table summarizes the evaluation tasks to be performed for the **RSA OAEP** asymmetric encryption construction:

Summary of Evaluation Tasks for RSA OAEP Encryption Scheme	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/AsymEncryption
	Underlying Primitive CCN-MECHANISM/RSA CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/AsymEncryption **Notes** Note 34 [Random Padding] Note 35 [Padding-OAEP Attack]
	Underlying Primitive CCN-AGREED/RSA CCN-AGREED/Hash
	RSA Key Pair Generation (if applicable) CCN-AGREED/RSAKeys **Notes** Note 29 [Primes] Note 61 [Primes from Uniform Distributions] Note 62 [Miller-Rabin as Primality Test] Note 63 [Miller-Rabin Parameters] Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 65)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymEncryption
	Underlying Primitive

Summary of Evaluation Tasks for RSA OAEP Encryption Scheme	
Evaluation Task	**Evaluation Test Code**
	CCN-PITFALL/RSA CCN-PITFALL/RSAParameter.1 CCN-PITFALL/RSAParameter.2

Table 100. Summary of Evaluation Tasks for RSA OAEP Asymmetric Encryption

319. The following table summarizes the evaluation tasks to be performed for the **RSA PKCS#1v1.5** asymmetric encryption construction:

Summary of Evaluation Tasks for RSA PKCS#1v1.5 Encryption Scheme	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/AsymEncryption
	Underlying Primitive CCN-MECHANISM/RSA
CCN-AGREED	**Construction** CCN-AGREED/AsymEncryption **Notes** Note 34 [Random Padding]
	Underlying Primitive CCN-AGREED/RSA
	RSA Key Pair Generation (if applicable) CCN-AGREED/RSAKeys **Notes** Note 29 [Primes] Note 61 [Primes from Uniform Distributions] Note 62 [Miller-Rabin as Primality Test] Note 63 [Miller-Rabin Parameters] Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 65)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymEncryption CCN-PITFALL/RSAPKCS

Summary of Evaluation Tasks for RSA PKCS#1v1.5 Encryption Scheme	
Evaluation Task	**Evaluation Test Code**
	Underlying Primitive CCN-PITFALL/RSA CCN-PITFALL/RSAParameter.1 CCN-PITFALL/RSAParameter.2

Table 101. Summary of Evaluation Tasks for RSA PKCS#1v1.5 Encryption Scheme

6.2.2 DIGITAL SIGNATURE

320. This table summarizes the evaluation tasks to be performed for Digital Signature constructions based on approved **RSA-PSS** schemes:

Summary of Evaluation Tasks for RSA-PSS Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/RSA CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function]
	Underlying Primitive CCN-AGREED/RSA CCN-AGREED/Hash
	RSA Key Pair Generation (if applicable) CCN-AGREED/RSAKeys **Notes** Note 29 [Primes] Note 61 [Primes from Uniform Distributions] Note 62 [Miller-Rabin as Primality Test] Note 63 [Miller-Rabin Parameters] Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)

Summary of Evaluation Tasks for RSA-PSS Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/RSA CCN-PITFALL/RSAParameter.1 CCN-PITFALL/RSAParameter.2

Table 102. Summary of Evaluation Tasks for RSA-PSS Digital Signature

321. This table summarizes the evaluation tasks to be performed for the Digital Signature constructions based on approved **KCDSA** schemes:

Summary of Evaluation Tasks for KCDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/FFDLOG CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 38 [DSA Randomness]
	Underlying Primitive CCN-AGREED/FFDLOG CCN-AGREED/Hash
	FFDLOG Key Pair Generation (if applicable) CCN-AGREED/FFDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction**

Summary of Evaluation Tasks for KCDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
	CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/FFDLOG.1 CCN-PITFALL/FFDLOG.2

Table 103. Summary of Evaluation Tasks for KCDSA Digital Signature

322. This table summarizes the evaluation tasks to be performed for the Digital Signature constructions based on approved **Schnorr** schemes:

Summary of Evaluation Tasks for Schnorr Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/FFDLOG CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 38 [DSA Randomness]
	Underlying Primitive CCN-AGREED/FFDLOG CCN-AGREED/Hash
	FFDLOG Key Pair Generation (if applicable) CCN-AGREED/FFDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth

Summary of Evaluation Tasks for Schnorr Digital Signature	
Evaluation Task	**Evaluation Test Code**
	Underlying Primitive CCN-PITFALL/FFDLOG.1 CCN-PITFALL/FFDLOG.2

Table 104. Summary of Evaluation Tasks for Schnorr Digital Signature

323. This table summarizes the evaluation tasks to be performed for the Digital Signature constructions based on approved **DSA** schemes:

Summary of Evaluation Tasks for DSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/FFDLOG CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 38 [DSA Randomness]
	Underlying Primitive CCN-AGREED/FFDLOG CCN-AGREED/Hash
	FFDLOG Key Pair Generation (if applicable) CCN-AGREED/FFDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive

Summary of Evaluation Tasks for DSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
	CCN-PITFALL/FFDLOG.1 CCN-PITFALL/FFDLOG.2

Table 105. Summary of Evaluation Tasks for DSA Digital Signature

324. This table summarizes the evaluation tasks to be performed for Digital Signature constructions based on approved **EC-KCDSA** schemes:

Summary of Evaluation Tasks for EC-KCDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/ECDLOG CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 38 [DSA Randomness]
	Underlying Primitive CCN-AGREED/ECDLOG CCN-AGREED/Hash **Notes** Note 30 [Points on the Curve] Note 31 [Points on a Subgroup] Note 32 [Prime Order] Note 33 [Prime Selection]
	ECDLOG Key Pair Generation (if applicable) CCN-AGREED/ECDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction

Summary of Evaluation Tasks for EC-KCDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
	CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/ECDLOG

Table 106. Summary of Evaluation Tasks for EC-KCDSA Digital Signature

325. This table summarizes the evaluation tasks to be performed for Digital Signature constructions based on approved **EC-DSA** schemes:

Summary of Evaluation Tasks for EC-DSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/ECDLOG CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 38 [DSA Randomness]
	Underlying Primitive CCN-AGREED/ECDLOG CCN-AGREED/Hash **Notes** Note 30 [Points on the Curve] Note 31 [Points on a Subgroup] Note 32 [Prime Order] Note 33 [Prime Selection]
	ECDLOG Key Pair Generation (if applicable) CCN-AGREED/ECDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)

Summary of Evaluation Tasks for EC-DSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/ECDLOG

Table 107. Summary of Evaluation Tasks for EC-DSA Digital Signature

326. This table summarizes the evaluation tasks to be performed for Digital Signature constructions based on approved **EC-GDSA** schemes:

Summary of Evaluation Tasks for EC-GDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/ECDLOG CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 38 [DSA Randomness]
	Underlying Primitive CCN-AGREED/ECDLOG CCN-AGREED/Hash **Notes** Note 30 [Points on the Curve] Note 31 [Points on a Subgroup] Note 32 [Prime Order] Note 33 [Prime Selection]
	ECDLOG Key Pair Generation (if applicable) CCN-AGREED/ECDLOGKeys **Notes** Note 60 [DRNG as Input]

Summary of Evaluation Tasks for EC-GDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/ECDLOG

Table 108. Summary of Evaluation Tasks for EC-GDSA Digital Signature

327. This table summarizes the evaluation tasks to be performed for Digital Signature constructions based on approved **EC-Schnorr** schemes:

Summary of Evaluation Tasks for EC-Schnorr Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/ECDLOG CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 38 [DSA Randomness]
	Underlying Primitive CCN-AGREED/ECDLOG CCN-AGREED/Hash **Notes** Note 30 [Points on the Curve] Note 31 [Points on a Subgroup] Note 32 [Prime Order] Note 33 [Prime Selection]
	ECDLOG Key Pair Generation (if applicable) CCN-AGREED/ECDLOGKeys

Summary of Evaluation Tasks for EC-Schnorr Digital Signature	
Evaluation Task	**Evaluation Test Code**
	Notes Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/ECDLOG

Table 109. Summary of Evaluation Tasks for EC-Schnorr Digital Signature

328. This table summarizes the evaluation tasks to be performed for Digital Signature constructions based on approved **EdDSA** schemes:

Summary of Evaluation Tasks for EdDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/ECDLOG CCN-MECHANISM/Hash CCN-MECHANISM/XOF
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature
	Underlying Primitive CCN-AGREED/ECDLOG CCN-AGREED/Hash CCN-AGREED/XOF **Notes** Note 30 [Points on the Curve] Note 31 [Points on a Subgroup] Note 32 [Prime Order] Note 3 [Agreed XOF]
	ECDLOG Key Pair Generation (if applicable)

Summary of Evaluation Tasks for EdDSA Digital Signature	
Evaluation Task	**Evaluation Test Code**
	CCN-AGREED/ECDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/ECDLOG

Table 110. Summary of Evaluation Tasks for EdDSA Digital Signature

329. This table summarizes the evaluation tasks to be performed for Digital Signature constructions based on approved **RSA-PKCS#1v1.5** schemes**:**

Summary of Evaluation Tasks for RSA PKCS#1v1.5 Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/RSA CCN-MECHANISM/Hash
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 36 [Hash Function] Note 37 [PKCS Format Check]
	Underlying Primitive CCN-AGREED/RSA CCN-AGREED/Hash
	RSA Key Pair Generation (if applicable) CCN-AGREED/RSAKeys **Notes** Note 29 [Primes]

Summary of Evaluation Tasks for RSA PKCS#1v1.5 Digital Signature	
Evaluation Task	**Evaluation Test Code**
	Note 61 [Primes from Uniform Distributions] Note 62 [Miller-Rabin as Primality Test] Note 63 [Miller-Rabin Parameters] Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth
	Underlying Primitive CCN-PITFALL/RSA CCN-PITFALL/RSAParameter.1 CCN-PITFALL/RSAParameter.2

Table 111. Summary of Evaluation Tasks for RSA PKCS#1v1.5 Digital Signature

330. The following table summarizes the evaluation tasks to be performed for the **XMSS** scheme based on **hash** and **XOF** primitives:

Summary of Evaluation Tasks for XMSS Digital Signature	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DigitalSignature
	Underlying Primitive CCN-MECHANISM/Hash CCN-MECHANISM/XOF
CCN-AGREED	**Construction** CCN-AGREED/DigitalSignature **Notes** Note 39 [XMSS Underlying Primitives] Note 40 [Stateful Post-quantum Digital Signatures1] Note 41 [Stateful Post-quantum Digital Signatures2] Note 42 [Precautions for XMSS Backups] Note 43 [Stateful Post-quantum Digital Signatures3] Note 44 [Stateful Post-quantum Digital Signatures4]
	Underlying Primitive

Summary of Evaluation Tasks for XMSS Digital Signature	
Evaluation Task	**Evaluation Test Code**
	CCN-AGREED/Hash CCN-AGREED/XOF **Notes** Note 3 [Agreed XOF]
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 66)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/AsymAuth CCN-PITFALL/XMSS.1 CCN-PITFALL/XMSS.2 CCN-PITFALL/XMSS.3 CCN-PITFALL/XMSS.4
	Underlying Primitive N/A because there are no implementation pitfalls for hash and XOF primitives

Table 112. Summary of Evaluation Tasks for XMSS Digital Signature

6.2.3 KEY ESTABLISHMENT

331. The following table summarizes the evaluation tasks to be performed for the **DH** key establishment constructions. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in section 6.1.9 "Key Derivation Functions" for the underlying key derivation cryptographic constructions.

Summary of Evaluation Tasks for DH Key Establishment Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyEstablishment
	Underlying Primitive CCN-MECHANISM/FFDLOG
CCN-AGREED	**Construction** CCN-AGREED/KeyEstablishment

Summary of Evaluation Tasks for DH Key Establishment Constructions	
Evaluation Task	**Evaluation Test Code**
	Notes Note 46 [Authentication] Note 47 [DH/EC-DH Subgroup Attacks] Note 50 [Uniform Distributions]
	Underlying Primitive CCN-AGREED/FFDLOG
	FFDLOG Key Pair Generation (if applicable) CCN-AGREED/FFDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 67)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/KeyEstablishment CCN-PITFALL/FFKeyEstablishment
	Underlying Primitive CCN-PITFALL/FFDLOG.1 CCN-PITFALL/FFDLOG.2

Table 113. Summary of Evaluation Tasks for DH Key Establishment Constructions

332. The following table summarizes the evaluation tasks to be performed for the **DLIES-KEM** constructions. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in section 6.1.9 "Key Derivation Functions" for the underlying key derivation cryptographic constructions.

Summary of Evaluation Tasks for DLIES-KEM Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyEstablishment
	Underlying Primitive CCN-MECHANISM/FFDLOG
CCN-AGREED	**Construction**

Summary of Evaluation Tasks for DLIES-KEM Constructions	
Evaluation Task	**Evaluation Test Code**
	CCN-AGREED/KeyEstablishment **Notes** Note 46 [Authentication] Note 47 [DH/EC-DH Subgroup Attacks] Note 48 [DLIES Key Length] Note 50 [Uniform Distributions]
	Underlying Primitive CCN-AGREED/FFDLOG
	FFDLOG Key Pair Generation (if applicable) CCN-AGREED/FFDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 67)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/KeyEstablishment CCN-PITFALL/FFKeyEstablishment
	Underlying Primitive CCN-PITFALL/FFDLOG.1 CCN-PITFALL/FFDLOG.2

Table 114. Summary of Evaluation Tasks for DLIES-KEM Constructions

333. The following table summarizes the evaluation tasks to be performed for **ECDH** key establishment constructions. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in section 6.1.9 "Key Derivation Functions" for the underlying key derivation cryptographic constructions.

Summary of Evaluation Tasks for ECDH Key Establishment Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyEstablishment

Summary of Evaluation Tasks for ECDH Key Establishment Constructions	
Evaluation Task	**Evaluation Test Code**
	Underlying Primitive CCN-MECHANISM/ECDLOG
CCN-AGREED	**Construction** CCN-AGREED/KeyEstablishment **Notes** Note 46 [Authentication] Note 47 [DH/EC-DH Subgroup Attacks] Note 50 [Uniform Distributions]
	Underlying Primitive CCN-AGREED/ECDLOG **Notes** Note 30 [Points on the Curve] Note 31 [Points on a Subgroup] Note 32 [Prime Order] Note 33 [Prime Selection]
	ECDLOG Key Pair Generation (if applicable) CCN-AGREED/ECDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 67)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/KeyEstablishment CCN-PITFALL/ECKeyEstablishment
	Underlying Primitive CCN-PITFALL/ECDLOG

Table 115. Summary of Evaluation Tasks for ECDH Key Establishment Constructions

334. The following table summarizes the evaluation tasks to be performed for **ECIES-KEM** constructions. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in section 6.1.9 "Key Derivation Functions" for the underlying key derivation cryptographic constructions.

Summary of Evaluation Tasks for ECIES-KEM Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/KeyEstablishment
	Underlying Primitive CCN-MECHANISM/ECDLOG
CCN-AGREED	**Construction** CCN-AGREED/KeyEstablishment **Notes** Note 46 [Authentication] Note 47 [DH/EC-DH Subgroup Attacks] Note 49 [ECIES Key Length] Note 50 [Uniform Distributions]
	Underlying Primitive CCN-AGREED/ECDLOG **Notes** Note 30 [Points on the Curve] Note 31 [Points on a Subgroup] Note 32 [Prime Order] Note 33 [Prime Selection]
	ECDLOG Key Pair Generation (if applicable) CCN-AGREED/ECDLOGKeys **Notes** Note 60 [DRNG as Input]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 67)
CCN-PITFALL	**Construction** CCN-PITFALL/AsymConstruction CCN-PITFALL/KeyEstablishment CCN-PITFALL/ECKeyEstablishment
	Underlying Primitive CCN-PITFALL/ECDLOG

Table 116. Summary of Evaluation Tasks for ECIES-KEM Constructions

6.3 RANDOM NUMBER GENERATION

6.3.1 TRUE RANDOM NUMBER GENERATORS

335. The following table summarizes the evaluation tasks to be performed for **physical true random number generators** (PTRNG):

Summary of Evaluation Tasks for Physical TRNGs	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**TRNG** CCN-MECHANISM/TRNG
CCN-AGREED	**PTRNG** CCN-AGREED/PTRNG **Notes** Note 52 [TRNG Use] Note 53 [PTRNG Evaluation]
	TRNG CCN-AGREED/TRNGEntropy

Table 117. Summary of Evaluation Tasks for PTRNGs

336. The following table summarizes the evaluation tasks to be performed for **non-physical true random number generators** (NPTRNG):

Summary of Evaluation Tasks for Non-Physical TRNGs	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**TRNG** CCN-MECHANISM/TRNG
CCN-AGREED	**NPTRNG** CCN-AGREED/NPTRNG **Notes** Note 52 [TRNG Use] Note 54 [NPTRNG Evaluation]
	TRNG CCN-AGREED/TRNGEntropy

Table 118. Summary of Evaluation Tasks for NPTRNGs

6.3.2 DETERMINISTIC RANDOM NUMBER GENERATORS

337. The following table summarizes the evaluation tasks to be performed for deterministic random number generators based on **HMAC-DRBG** [SP800-90A] construction. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

    - The **tester shall** perform the evaluation tasks defined in Table 81 for the HMAC underlying constructions.

Summary of Evaluation Tasks for HMAC-DRBG Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DRNG
CCN-AGREED	**Construction** CCN-AGREED/DRNG CCN-AGREED/DRNGScheme **Notes** Note 55 [DRNG Evaluation] Note 56 [DRNG HMAC-DRBG Conformity Proof] Note 58 [DRNG Seeding] Note 59 [DRNG Backtracking Resistance]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 68)
CCN-PITFALL	**Construction** CCN-PITFALL/DRG.1 CCN-PITFALL/DRG.2

Table 119. Summary of Evaluation Tasks for HMAC-DRBG Constructions

338. The following table summarizes the evaluation tasks to be performed for deterministic random number generators based on **Hash-DRBG** [SP800-90A] construction.

Summary of Evaluation Tasks for Hash-DRBG Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DRNG
	Underlying Primitive CCN-MECHANISM/Hash
CCN-AGREED	**Construction**

Summary of Evaluation Tasks for Hash-DRBG Constructions	
Evaluation Task	**Evaluation Test Code**
	CCN-AGREED/DRNG CCN-AGREED/DRNGScheme **Notes** Note 55 [DRNG Evaluation] Note 57 [DRNG Hash-DRBG Conformity Proof] Note 58 [DRNG Seeding] Note 59 [DRNG Backtracking Resistance]
	Underlying Primitive CCN-AGREED/Hash
CCN-CONFORMITY	**Construction and Underlying Primitive** CCN-CONFORMITY/Mechanisms (Table 68)
CCN-PITFALL	**Construction** CCN-PITFALL/DRG.1 CCN-PITFALL/DRG.2
	Underlying Primitive N/A because there are no implementation pitfalls for the SHA cryptographic primitive

Table 120. Summary of Evaluation Tasks for Hash-DRBG Constructions

339. The following table summarizes the evaluation tasks to be performed for deterministic random number generators based on **CTR-DRBG** [SP800-90A] construction. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in Table 76 for the CTR underlying symmetric encryption constructions.

Summary of Evaluation Tasks for CTR-DRBG Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DRNG
CCN-AGREED	**Construction** CCN-AGREED/DRNG CCN-AGREED/DRNGScheme **Notes** Note 55 [DRNG Evaluation]

Summary of Evaluation Tasks for CTR-DRBG Constructions	
Evaluation Task	**Evaluation Test Code**
	Note 58 [DRNG Seeding] Note 59 [DRNG Backtracking Resistance]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table 68)
CCN-PITFALL	**Construction** CCN-PITFALL/DRG.1 CCN-PITFALL/DRG.2 CCN-PITFALL/CTRDRBG

Table 121. Summary of Evaluation Tasks for CTR-DRBG Constructions

340. In case of any other DRNG construction, the following table summarizes the evaluation tasks to be performed. It is mandatory to perform the evaluation tasks associated with the underlying constructions:

- The **tester shall** perform the evaluation tasks defined in the tables corresponding to the appropriate underlying constructions.

Summary of Evaluation Tasks for Generic DRNG Constructions	
Evaluation Task	**Evaluation Test Code**
CCN-MECHANISM	**Construction** CCN-MECHANISM/DRNG
CCN-AGREED	**Construction** CCN-AGREED/DRNG **Notes** Note 55 [DRNG Evaluation]
CCN-CONFORMITY	**Construction and Underlying Mechanisms** CCN-CONFORMITY/Mechanisms (Table depends on DRNG construction)
CCN-PITFALL	**Construction** CCN-PITFALL/DRG.1 CCN-PITFALL/DRG.2

Table 122. Summary of Evaluation Tasks for Generic DRNG Constructions

6.4 GENERAL EVALUATION TASKS

341. This section contains general evaluation tasks, related to the management of SSPs, the execution of cryptographic mechanisms self-tests, the mitigation of other attacks, and the implementation of cryptographic protocols, as part of the evaluation of cryptographic mechanisms. The evaluation tasks shall be performed depending on the usage or implementation of the cryptographic mechanisms.

6.4.1 SENSITIVE SECURITY PARAMETER MANAGEMENT

342. The following table summarizes the evaluation tasks to be performed for the **sensitive security parameter management**:

Summary of Evaluation Tasks for SSP Management	
Evaluation Task	**Evaluation Test Code**
CCN-SSP	CCN-SSP/Generation
	CCN-SSP/Transport
	CCN-SSP/Storage
	CCN-SSP/Zeroization.1
	CCN-SSP/Zeroization.2
CCN-PITFALL	CCN-PITFALL/KeyManagement
	CCN-PITFALL/KeyTransport
	CCN-PITFALL/KeyUsage

Table 123. Summary of Evaluation Tasks for SSP Management

6.4.2 CRYPTOGRAPHIC MECHANISMS SELF-TEST

343. The following table summarizes the evaluation tasks to be performed for the **implementation of cryptographic mechanisms self-test**:

Summary of Evaluation Tasks for Self-Test Implementation	
Evaluation Task	**Evaluation Test Code**
CCN-SELFTEST	CCN-SELFTEST/Implementation

Table 124. Summary of Evaluation Tasks for Self-Test Implementation

6.4.3 MITIGATION OF OTHER ATTACKS

344. The following table summarizes the evaluation tasks to be performed for the **mitigation of other attacks**:

Summary of Evaluation Tasks for the Mitigation of Other Attacks	
Evaluation Task	**Evaluation Test Code**
CCN-MITIGATION	CCN-MITIGATION/Mitigation

Table 125. Summary of Evaluation Tasks for the Mitigation of Other Attacks

6.4.4 CRYPTOGRAPHIC PROTOCOLS IMPLEMENTATION

345. The following table summarizes the evaluation tasks to be performed for the **cryptographic protocols implementation**. It is mandatory to perform the evaluation tasks associated with the underlying cryptographic mechanisms.

Summary of Evaluation Tasks for the Cryptographic Protocols	
Evaluation Task	**Evaluation Test Code**
CCN-PROTOCOL	**Protocol Implementation** CCN-PROTOCOL/TLS **Notes** Note 64 [psk_ke mode] Note 65 [0-RTT Data] Note 66 [TLS Encrypt and then MAC] Note 67 [TLS with RSA]
	Protocol Implementation CCN-PROTOCOL/SSH **Notes** Note 68 [Diffie-Hellman-group-exchange-SHA256] Note 69 [Key Renewal] Note 70 [ECDH-SHA2-*] Note 71 [Security Statement] Note 72 [ECDSA-SHA2-*] Note 73 [x509v3-ECDSA-SHA2-*]
	Protocol Implementation CCN-PROTOCOL/IPsec **Notes** Note 74 [Tunnel Mode – Transport Mode] Note 75 [IPsec Attacks] Note 76 [HMAC-SHA-XXX-YYY] Note 77 [RSASSA-PSS]

Table 126. Summary of Evaluation Tasks for Cryptographic Protocols

7 ANNEX B: CONFORMITY TESTS SPECIFICATION

346. This annex contains the definition of the test vectors of the conformity testing process. It is located in the external document [CCN-MEMeC-B].

8 REFERENCES

[ANS11] ANSSI. Avis relatif aux paramètres de courbes ellitpiques définis pas l'État français. Agence Nationale de la Sécurité des Systèmes d'Information, Journal O-ciel 0241 (Oct. 2011), p. 17533, 2011.

[ANSI-X9.63] ANSI. Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography (R2017). American National Standards Institute, ANSI X9.63:2011, 2017.

[BN00] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Proc. International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology - ASIACRYPT 2000, Lecture Notes Comput. Sci., volume 1976, pages 531-545, 2000.

[CCN-MEMeC-B] CCN Cryptographic Mechanisms Evaluation Methodology Annex B: Conformity Tests Specification v1.3.3. November 2024.

[CCN-STIC-221] CCN. Guía de Mecanismos Criptográficos autorizados por el CCN v2.1. Octubre 2024.

[FCRNG] BSI. A Proposal for Functionality Classes for Random Number Generators, version v3.0, 2024.

[FIPS-180-4] NIST. Digital Signature Standard (DSS). National Institute of Standard and Technology, NIST FIPS 186-4, 2013.

[FIPS-186-4] NIST. Digital Signature Standard (DSS). National Institute of Standard and Technology, NIST FIPS 186-4, 2013.

[FIPS-186-5] NIST. Digital Signature Standard (DSS). National Institute of Standard and Technology, NIST FIPS 186-5, 2023.

[FIPS-197] NIST. Advanced Encryption Standard. National Institute of Standard and Technology, Federal Information Processing Standard Publication, FIPS 197, 2001.

[FIPS-202] NIST. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. National Institute of Standard and Technology, NIST FIPS 202, 2015.

[FIPS-203] NIST. Module-Lattice-Based Key-Encapsulation Mechanism Standard. National Institute of Standard and Technology, NIST FIPS 203, 2024.

[FIPS-204] NIST. Module-Lattice-Based Digital Signature Standard. National Institute of Standard and Technology, NIST FIPS 204, 2024.

[FIPS-205] NIST. Stateless Hash-Based Digital Signature Standard. National Institute of Standard and Technology, NIST FIPS 205, 2024.

[ISO-10116] ISO/IEC 10116. Information technology – Security techniques – Modes of operation for an n-bit block cipher. International Organization for Standardization/International Electrotechnical Commission, 2017.

[ISO-10118-3] ISO/IEC 10118-3. ISO/IEC 10118-3:2018 – Information technology Security techniques – Hash-functions – Part 3: Dedicated hash-functions. ISO/IEC10118-3:2018. International Organization for Standardization/International Electrotechnical Commission, 2018.

[ISO-11770-3] ISO/IEC 11770-3. Information technology – Security techniques – Key management, Part 3: Mechanisms using asymmetric techniques, ISO/IEC 11770-3. International Organization for Standardization, 2015.

[ISO-14888-3] ISO/IEC 14888-3. Information technology – Security techniques – Digital signatures with appendix, Part 3: Discrete logarithm based mechanisms, ISO/IEC 14888-3. International Organization for Standardization, 2018.

[ISO-18031] ISO/IEC 18031. Information Technology – Security Techniques – Random bit generation, ISO/IEC 18031. International Organization for Standardization/International Electrotechnical Commission, 2011.

[ISO-18033-2] ISO/IEC 18033-2. Information Technology – Security Techniques – Encryption Algorithms-Part 2: Asymmetric Ciphers. International Organization for Standardization/International Electrotechnical Commission, 2006.

[ISO-18033-3] ISO/IEC. Information Technology – Security Techniques – Encryption Algorithms – Part 3: Block Ciphers. International Organization for Standardization/International Electrotechnical Commission, 2010.

[ISO-19772] ISO/IEC 19772. Information Technology – Authenticated encryption. International Organization for Standardization/International Electrotechnical Commission, 2020.

[ISO-9796-2] ISO/IEC 9796-2. Information technology Security techniques – Digital signature schemes giving message recovery, Part 2: Integer factorization based mechanisms, ISO/IEC 9796-2. International Organization for Standardization, 2010.

[ISO-9797-1] ISO/IEC 9797-1. Information technology Security techniques – Message Authentication Codes (MACs), Part 1: Mechanisms using a block cipher, ISO/IEC 9797-1. International Organization for Standardization/International Electrotechnical Commission, 2011.

[ISO-9797-2] ISO/IEC 9797-2. Information technology Security techniques – Message Authentication Codes (MACs), Part 2: Mechanisms using a dedicated hash-function. International Organization for Standardization/International Electrotechnical Commission, 2011.

[JSS15] "Practical Invalid Curve Attacks on TLS-ECDH", J. Jager, J. Schwenk, J. Somorovsky, ESORICS'15.

[Man01] "A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0", J. Manger, CRYPTO'01.

[PFH+20] T. Prest, P.-A. Fouque, J. Hostein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang.FALCON. Online publication, 2020. https://falcon-sign.info/.

[RFC-2104] HMAC: Keyed-Hashing for Message Authentication. Internet Engineering Task Force, RFC 2104, 1997.

[RFC-3526] More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE). NetworkWorking Group, 2003.

[RFC-4055] Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Internet Engineering Task Force, RFC 4055, 2005.

[RFC-4251] The Secure Shell (SSH) Protocol Architecture. Internet Engineering Task Force, RFC 4251, 2006.

[RFC-4253] The Secure Shell (SSH) Transport Layer Protocol. Internet Engineering Task Force, RFC 4253, 2006.

[RFC-4303] IP Encapsulating Security Payload (ESP). Internet Engineering Task Force, RFC 4303, 2005.

[RFC-4344] The Secure Shell (SSH) Transport Layer Encryption Modes. Internet Engineering Task Force, RFC 4344, 2006.

[RFC-4419] Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol. Internet Engineering Task Force, RFC 4419, 2006.

[RFC-4494] The AES-CMAC-96 Algorithm and Its Use with IPsec. Internet Engineering Task Force, RFC 4494, 2006.

[RFC-4543] Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specication. Internet Engineering Task Force, RFC 4543, 2006.

[RFC-4615] The Advanced Encryption Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE). Internet Engineering Task Force, RFC 4615, 2006.

[RFC-4754] IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA). Internet Engineering Task Force, RFC 4754, 2007.

[RFC-4868] Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec. Internet Engineering Task Force, RFC 4868, 2007.

[RFC-5246] The Transport Layer Security (TLS) Protocol, Version 1.2. Internet Engineering Task Force, RFC 5246, 2008.

[RFC-5288] AES Galois Counter Mode (GCM) Cipher Suites for TLS. Internet Engineering Task Force, RFC 5288, 2008.

[RFC-5289] TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM). Internet Engineering Task Force, RFC 5289, 2008.

[RFC-5297] Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES). Internet Engineering Task Force, RFC 5297, 2008.

[RFC-5487] Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode. Internet Engineering Task Force, RFC 5487, 2009.

[RFC-5639] Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve generation. Internet Engineering Task Force, RFC 5639, 2010.

[RFC-5647] AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. Internet Engineering Task Force, RFC 5647, 2009.

[RFC-5656] Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer. Internet Engineering Task Force, RFC 5656, 2009.

[RFC-5869] HMAC-based Extract-and-Expand Key Derivation Function (HKDF). Internet Engineering Task Force, RFC 5689, 2010.

[RFC-5903] Elliptic Curve Groups modulo a Prime (ECP Groups) for IKE and IKEv2. Internet Engineering Task Force, RFC 5903, 2010.

[RFC-6187] X.509v3 Certicates for Secure Shell Authentication. Internet Engineering Task Force, RFC 6187, 2011.

[RFC-6655] AES-CCM Cipher Suites for Transport Layer Security TLS. Internet Engineering Task Force, RFC 6655, 2008.

[RFC-6954] Using the Elliptic Curve Cryptography (ECC) Brainpool Curves for the Internet Key Exchange Protocol Version 2 (IKEv2). Internet Engineering Task Force, RFC 6954, 2013.

[RFC-7027] Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS). Internet Engineering Task Force, RFC 7027, 2013.

[RFC-7251] AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS. Internet Engineering Task Force, RFC 7251, 2014.

[RFC-7296] Internet Key Exchange Protocol Version 2 (IKEv2). Internet Engineering Task Force, RFC 7296, 2014.

[RFC-7427] Signature Authentication in the Internet Key Exchange Version 2 (IKEv2). Internet Engineering Task Force, RFC 7427, 2015.

[RFC-7693] The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC). Internet Engineering Task Force, RFC 7693, 2015.

[RFC-7748] Elliptic curves for security. Internet Engineering Task Force, RFC 7748, 2016.

[RFC-7905] ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS). Internet Engineering Task Force, RFC 7905, 2016.

[RFC-7914] The SCRYPT Password-Based Key Derivation Function. Internet Engineering Task Force, RFC 7914, 2016.

[RFC-7919] Negotiated Finite Field Di-e-Hellman Ephemeral Parameters for Transport Layer Security (TLS). Internet Engineering Task Force, RFC 7919, 2016.

[RFC-8017] PKCS #1: RSA Cryptography Specications Version 2.2. Internet Engineering Task Force, RFC 8017, 2016.

[RFC-8018] PKCS #5: Password-Based Cryptography Specication. Version 2.1. Internet Engineering Task Force, RFC 8018, 2017.

[RFC-8031] Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement. Internet Engineering Task Force, RFC 8031, 2016.

[RFC-8032] Edwards-Curve Digital Signature Algorithm (EdDSA). Internet Engineering Task Force, RFC 8032, 2017.

[RFC-8221] Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH). Internet Engineering Task Force, RFC 8221, 2017.

[RFC-8247] Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2). Internet Engineering Task Force, RFC 8247, 2017.

[RFC-8391] XMSS: extended Merkle signature scheme. Internet Engineering Task Force, RFC 8391, 2018.

[RFC-8422] Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier. Internet Engineering Task Force, RFC 8422, 2013.

[RFC-8439] ChaCha20 and Poly1305 for IETF Protocols. Internet Engineering Task Force, RFC 8439, 2018.

[RFC-8446] The Transport Layer Security (TLS) Protocol Version 1.3. Internet Engineering Task Force, RFC 8446, 2018.

[RFC-8734] Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS) Version 1.3. Internet Engineering Task Force, RFC 8734, 2020.

[RFC-9106] Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications. Internet Engineering Task Force, RFC 9106, 2021.

[RSA78] R. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21(2):120-126, 1978.

[Sha79] A. Shamir. How to share a secret. Communications of the ACM, 22(11):612-613, 1979.

[SOGIS-HEP] SOG-IS Crypto Evaluation Scheme Harmonised Cryptographic Evaluation Procedures v0.16. December 2020.

[SP800-108] NIST. Recommendation for Key Derivation Using Pseudorandom Functions. National Institute of Standard and Technology, Special Publication SP800-108, Revision 1, 2022.

[SP800-132] NIST. Recommendation for Password-Based Key Derivation. Part 1: Storage Applications. National Institute of Standard and Technology, Special Publication SP800-132, Addendum to SP800-132, 2010.

[SP800-133] NIST. Recommendation for Cryptographic Key Generation. National Institute of Standard and Technology, Special Publication SP800-133, Revision 2, 2020.

[SP800-185] NIST. SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash. National Institute of Standard and Technology, Special Publication, SP800-185, Rev 1 (November 2014 draft), 2016.

[SP800-186] NIST. Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters. National Institute of Standard and Technology, Special Publication, SP800-186 (February 2022), 2023.

[SP800-208] NIST. Recommendation for Stateful Hash-Based Signature Schemes. National Institute of Standard and Technology, Special Publication, SP800-208, 2020.

[SP800-38A] NIST. Recommendation for Block Cipher Modes of Operation. Methods and Techniques. National Institute of Standards and Technology, Special Publication SP800-38A, March 2001.

[SP800-38A-Addendum] NIST. Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. National Institute of Standard and Technology, Special Publication, Addendum to SP800-38A, 2012.

[SP800-38B] NIST. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. National Institute of Standards and Technology, Special Publication SP800-38B, May 2005.

[SP800-38C] NIST. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. National Institute of Standards and Technology, Special Publication SP800-38C, May 2007.

[SP800-38D] NIST. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology, Special Publication SP800-38D, November 2007.

[SP800-38E] NIST. Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Condentiality on Storage Devices. National Institute of Standards and Technology, Special Publication SP800-38E, November 2010.

[SP800-38F] NIST. Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping. National Institute of Standards and Technology, Special Publication SP800-38F, December 2012.

[SP800-56A] NIST. Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography. National Institute of Standard and Technology, Special Publication, SP800-56A, Rev. 3, 2018.

[SP800-56B] NIST. Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography. National Institute of Standard and Technology, Special Publication, SP800-56B, Rev. 2, 2019.

[SP800-56C] NIST. Recommendation for Key-Derivation Methods in Key Establishment Schemes. National Institute of Standard and Technology, Special Publication, SP800-56C, Rev. 2, 2020.

[SP800-90A] NIST. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. National Institute of Standard and Technology, Special Publication, SP800-90A, Rev. 1, 2015.

[TR-03111] BSI. Elliptic curve cryptography. Bundesamt für Sicherheit in der Informationstechnik, BSI TR-03111 version 2.0, 2012.

[Vau02] “Security Flaws Induced by CBC Padding Applications to SSL, IPsec, WTLS, …”, S. Vaudenay, EUROCRYPT'02.

9 ABBREVIATIONS

AES	Advanced Encryption Scheme
CBC	Cipher Block Chaining
CCM	Counter with Cipher Block Chaining MAC
CCN	Centro Criptológico Nacional
CFB	Cipher Feedback
CL1	Certification Level 1
CL2	Certification Level 2
CL3	Certification Level 3
CRT	Chinese Remainder Theorem
CSP	Critical Security Parameter
CTR	Counter
DRBG	Deterministic Random Bit Generator
DRNG	Deterministic Random Number Generator
DSA	Digital Signature Mechanism
EAX	Encrypt-then-Authenticate-then-Translate
EC	Elliptic Curve
ECDLOG	Discrete Logarithm in Elliptic Curves
EC-DSA	Elliptic Curve Digital Signature Scheme
EC-FSDSA	Elliptic Curve Full Schnorr Signature Scheme
EC-GDSA	Elliptic Curve German Digital Signature Scheme
EC-KCDSA	Elliptic Curve Korean Certificate-based Digital Signature Scheme
EC-SDSA	Elliptic Curve Schnorr Signature Scheme
EdDSA	Edwards-curve Digital Signature Scheme
ENS	Esquema Nacional de Seguridad.
FFDLOG	Discrete Logarithm in Finite Fields
GCM	Galois/Counter
GMAC	Galois Message Authentication Code
HMAC	Hash-Based Message Authentication Code
I1	Cryptographic Mechanisms Evaluation Vendor Questionnaire and/or Vendor Documentation
I2	Test and operation interfaces to verify the TOE functionality
I3	Conformity testing tools
I4	RSPs obtained from the conformity testing
I5	Evidence of the avoidance of implementation pitfalls
I6	Implementation representation
I7	Self-Test input vectors
I8	Random Number Generation Vendor Questionnaire
IC	Integrated Circuit
KAT	Known Answer Test
KCDSA	Korean Certificate-based Digital Signature Mechanism
MAC	Message Authentication Code
MCT	Monte Carlo Test
NPTRNG	Non-Physical True Random Number Generator

OFB	Output Feedback
PSP	Public Security Parameter
PTRNG	Physical True Random Number Generator
RNG	Random Number Generator
RSA	Rivest-Shamir-Adleman Mechanism
SSP	Sensitive Security Parameter
TOE	Target Of Evaluation
TRNG	True Random Number Generator
VAL	Validation Test
XOF	Extended-Output Functions

PUBLICACIONES DE DEFENSA

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA
SUBSECRETARÍA DE DEFENSA
SECRETARÍA GENERAL TÉCNICA
SUBDIRECCIÓN GENERAL DE PUBLICACIONES Y PATRIMONIO CULTURAL