

Guía de Seguridad de las TIC

CCN-STIC 2100-Anexo A

Cryptographic Mechanisms Evaluation Vendor Questionnaire



Noviembre de 2024



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

Edita:



Pº de la Castellana 109, 28046 Madrid
© Centro Criptológico Nacional, 2024

Fecha de Edición: noviembre de 2024
NIPO: 083-25-057-3.



CENTRO CRIPTOLOGICO
NACIONAL
cn=CENTRO CRIPTOLOGICO
NACIONAL, 2.5.4.97=VATES-
S2800155J, ou=CENTRO
CRIPTOLOGICO NACIONAL,
o=CENTRO CRIPTOLOGICO
NACIONAL, c=ES
2025.05.27 10:03:13 +02'00'

LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

AVISO LEGAL

Quedan rigurosamente prohibidas, sin la autorización escrita del Centro Criptológico Nacional, bajo las sanciones establecidas en las leyes, la reproducción parcial o total de este documento por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares del mismo mediante alquiler o préstamo públicos.

TABLE OF CONTENTS

2	
1 INTRODUCTION.....	6
1.1 OBJECTIVE	6
1.2 STRUCTURE OF THE DOCUMENT	6
2 TOE DEFINITION.....	8
3 CRYPTOGRAPHIC MECHANISMS IMPLEMENTATION	10
3.1 SYMMETRIC ELEMENTARY PRIMITIVES	10
3.1.1 BLOCK CIPHER.....	10
3.1.2 STREAM CIPHER.....	11
3.1.3 HASH FUNCTIONS.....	12
3.1.4 SECRET SHARING	13
3.1.5 EXTENDABLE-OUTPUT FUNCTIONS (XOF)	14
3.2 SYMMETRIC CONSTRUCTIONS.....	16
3.2.1 SYMMETRIC ENCRYPTION (CONFIDENTIALITY ONLY)	16
3.2.2 DISK ENCRYPTION.....	18
3.2.3 INTEGRITY MODES: MESSAGE AUTHENTICATION CODE	19
3.2.4 SYMMETRIC ENTITY AUTHENTICATION SCHEMES.....	22
3.2.5 AUTHENTICATED ENCRYPTION (AE).....	23
3.2.6 KEY PROTECTION	26
3.2.7 KEY DERIVATION FUNCTIONS (KDF)	28
3.2.8 PASSWORD PROTECTION/HASHING MECHANISMS	33
3.3 ASYMMETRIC ELEMENTARY PRIMITIVES	36
3.3.1 RSA/INTEGER FACTORIZATION.....	36
3.3.2 MULTIPLICATIVE DISCRETE LOGARITHM PROBLEM (FFDLOG)	37
3.3.3 ADDITIVE DISCRETE LOGARITHM PROBLEM (ECDLOG)	38
3.4 ASYMMETRIC CONSTRUCTIONS.....	42
3.4.1 ASYMMETRIC ENCRYPTION SCHEMES	42
3.4.2 DIGITAL SIGNATURE	44
3.4.3 KEY ESTABLISHMENT	51
3.5 RANDOM NUMBER GENERATORS	55
3.5.1 TRUE RANDOM NUMBER GENERATORS	55
3.5.2 DETERMINISTIC RANDOM NUMBER GENERATOR CONSTRUCTION	56
4 DETERMINISTIC RANDOM NUMBER IMPLEMENTATION.....	59
4.1 GENERAL CONSIDERATIONS	59
4.2 DRNG INITIALIZATION	59
4.3 DRNG RESEEDING CONFIGURATION.....	60
5 SENSITIVE SECURITY PARAMETERS MANAGEMENT	61
5.1 SENSITIVE SECURITY PARAMETER GENERATION AND ESTABLISHMENT.....	61
5.2 SENSITIVE SECURITY PARAMETER ENTRY/OUTPUT	65
5.3 SENSITIVE SECURITY PARAMETER STORAGE	66
5.4 SENSITIVE SECURITY PARAMETER ZEROIZATION.....	67
6 CRYPTOGRAPHIC MECHANISMS SELF-TEST	69
7 MITIGATION OF OTHER ATTACKS	70

8 CRYPTOGRAPHIC PROTOCOLS IMPLEMENTATION	71
8.1 TRANSPORT LAYER SECURITY PROTOCOL (TLS)	72
8.1.1 TLS V1.3	72
8.1.2 TLS V1.2	74
8.2 SECURE SHELL (SSH)	78
8.3 INTERNET PROTOCOL SECURITY (IPSEC) WITH IKEV2.....	80
9 ANNEX A: SENSITIVE SECURITY PARAMETERS DEFINITION	83
10 ANNEX B: LIST OF SSPS MANAGED BY THE TOE.....	87
11 ANNEX C: LIST OF TEMPORALLY SENSITIVE DATA MANAGED BY THE TOE	90
12 REFERENCES.....	116
13 ACRONYMS.....	117

Note:

This document is an official reference document as part of the [CCN-MEMeC] methodology, so the **vendor shall** complete the vendor-specific version of this document.

1 INTRODUCTION

1.1 OBJECTIVE

1. This document is a vendor questionnaire that aims to gather the required information related to a cryptographic module implementation in order to comply with the CCN Cryptographic Mechanisms Evaluation Methodology [CCN-MEMeC].
2. On the one hand, this document will help vendors provide information about their cryptographic implementation with guided questions regarding the cryptographic mechanisms and the management of their associated CSPs and sensitive data. On the other hand, this document also will help the tester to gather all the needed information about the TOE cryptographic implementation by reducing the evaluation effort and time of reviewing.

Note: the **vendor shall** provide information on all cryptographic mechanisms implemented by the TOE, even if they are not in use.

3. Taking into account that this document contains the main topics related to the cryptographic mechanism implementation according to the [CCN-MEMeC] methodology, it will be useful for vendors to avoid inconsistencies or forget to include relevant information.

1.2 STRUCTURE OF THE DOCUMENT

4. The document is composed of a set of tables including the necessary questions to help the vendor specify the information that will be required by the lab in order to carry out the CCN Cryptographic Mechanisms Evaluation.
5. The document is organized with a structure similar to the main sections of the [CCN-MEMeC] methodology and includes a table of questions for each type of cryptographic mechanism (usually associated with a cryptographic test). Therefore, this will help the tester to associate the vendor information with the specific evaluation tasks.
6. The specific structure is as detailed below:
 - The name of the question tables will be the same as their associated section identifier, followed by the vendor questionnaire identifier, represented in the following format: **SectionID-VQ**.
 - Each table contains one or several questions related to vendor requirements and evaluation tests that will be identified with the same number as the question number indicated under the column name **Q-ID**.
7. The format of the table is as detailed below:

Q-ID	SectionID-VQ
1	Description of Question 1
	Row for vendor Answer 1
2	Description of Question 2
	Row for vendor Answer 2
...	...
	...
N	Description of Question N
	Row for vendor Answer N

8. **Note:** The tables that do not comply with this format are those included in the section below to provide the TOE definition and those included in the annexes, that are related to the SSPs management.

2 TOE DEFINITION

9. This section contains information related to the Target of Evaluation (TOE). The **vendor shall** provide the hardware, firmware, and/or software versions of the TOE and a brief description of its functionality from the cryptographic operation point of view.

Q-ID	TOE Description		
1	TOE Name The vendor shall provide the name/model of the TOE under evaluation.		
2	MEMEC Certification Level The vendor shall indicate the certification level (CL1, CL2, or CL3).		
3	TOE Versioning Information The vendor shall provide the version of the TOE under evaluation. Note: Indicate (N/A) if hardware, firmware, or software versions do not apply.		
	<table border="1"> <tr> <td>TOE Version</td><td></td></tr> </table>	TOE Version	
	TOE Version		
	<table border="1"> <tr> <td>Hardware Version</td><td></td></tr> </table>	Hardware Version	
	Hardware Version		
<table border="1"> <tr> <td>Firmware Version</td><td></td></tr> </table>	Firmware Version		
Firmware Version			
<table border="1"> <tr> <td>Software Version</td><td></td></tr> </table>	Software Version		
Software Version			
4	TOE Description The vendor shall provide a general description specifying the cryptographic capabilities of the TOE. This general description shall include at least the following information: - Description of the TOE, indicating its main cryptographic functionalities. - Descriptive figure of the TOE, indicating the cryptographic boundary that will be subject to evaluation. - List of all the cryptographic mechanisms included in each cryptographic implementation of the TOE. The purpose of each cryptographic mechanism implemented in the TOE shall be detailed in the tables associated with each of them.		
	<i>[TOE Description]</i> <i>[TOE Figure]</i> <i>[List of all the cryptographic mechanisms for each implementation]</i>		

Q-ID	TOE Description	
	Cryptographic Mechanism	Implementation

3 CRYPTOGRAPHIC MECHANISMS IMPLEMENTATION

10. This section contains the questions associated with the cryptographic mechanisms implemented in the TOE. It aims to provide the tester with all the needed information to demonstrate that the implementation of the cryptographic primitives and cryptographic constructions complies with the requirements from Sections 2 “CCN Cryptographic Requirements” and 3 “CCN Agreed Cryptographic Mechanisms” of the [CCN-MEMeC] methodology.

3.1 SYMMETRIC ELEMENTARY PRIMITIVES

3.1.1 BLOCK CIPHER

Q-ID	BlockCipher-VQ
1	Does the TOE implement <i>Block Cipher</i> cryptographic primitives? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Block Cipher</i> cryptographic primitives implemented in the TOE?
	☐ AES-128 ☐ AES-192 ☐ AES-256 ☐ Other(s) - Please indicate which one(s)
3	Does the TOE implement a self-test KAT for each implemented <i>Block Cipher</i> cryptographic primitive (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each block cipher cryptographic primitive (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).
4	Does the TOE implement a self-test KAT for each implemented key length (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key length (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).

Q-ID	BlockCipher-VQ
5	Does the TOE implement a self-test KAT for each implemented direction (ciphering and deciphering) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for ciphering and deciphering operations (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).
6	Please provide a brief description of the purpose of each implemented <i>Block Cipher</i> cryptographic primitive with its associated key length. If there is more than one implementation for the same primitive, please provide the purpose of each of them (on different rows). (e.g., AES-128 is implemented in the TOE as a primitive of the AES-128-GCM cryptographic mechanism of the cipher suites utilized by TLS v1.3) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.1.2 STREAM CIPHER

Q-ID	StreamCipher-VQ
1	Does the TOE implement <i>Stream Cipher</i> cryptographic primitives? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Stream Cipher</i> cryptographic primitives implemented in the TOE? ☐ ChaCha20
	☐ Other(s) - Please indicate which one(s)
3	Does the TOE implement a self-test KAT for each implemented <i>Stream Cipher</i> cryptographic primitive (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each stream cipher cryptographic primitive

Q-ID	StreamCipher-VQ
	(e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).
4	Does the TOE implement a self-test KAT for each implemented direction (ciphering and deciphering) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for ciphering and deciphering operations (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).
5	Please provide a brief description of the purpose of each implemented <i>Stream Cipher</i> cryptographic primitive. If there is more than one implementation for the same primitive, please provide the purpose of each of them (on different rows). (e.g., ChaCha20 is implemented in the TOE as a primitive of the ChaCha20-Poly1305 cryptographic mechanism of the cipher suites utilized by TLS v1.3) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.1.3 HASH FUNCTIONS

Q-ID	Hash-VQ
1	Does the TOE implement <i>Hash</i> cryptographic primitives? Note: If the response is “No”, the following questions in this table shall not be answered. ☐ Yes ☐ No
2	What are the <i>Hash</i> cryptographic primitives implemented in the TOE? ☐ SHA2-256 ☐ SHA2-384 ☐ SHA2-512 ☐ SHA2-512/256 ☐ SHA3-256 ☐ SHA3-384 ☐ SHA3-512 ☐ BLAKE2b ☐ Other(s) - Please indicate which one(s)

Q-ID	Hash-VQ
3	Does the TOE implement a self-test KAT for each implemented <i>Hash</i> cryptographic primitive (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each hash cryptographic primitive (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).
4	Please provide a brief description of the purpose of each implemented <i>Hash</i> cryptographic primitive. If there is more than one implementation for the same primitive, please provide the purpose of each of them (on different rows). (e.g., All the CSPs stored in the TOE are encrypted with AES-256-KeyWrapping, moreover, their SHA-256 digests are also stored in order to protect their integrity) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.1.4 SECRET SHARING

Q-ID	SecretSharing-VQ
1	Does the TOE implement the <i>Secret Sharing (Shamir)</i> cryptographic primitive? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	Does the TOE implement a self-test KAT for the implemented <i>Secret Sharing</i> recombination and generation operations (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for the secret sharing recombination and generation operations (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).

Q-ID	SecretSharing-VQ
3	Please provide a brief description of the purpose of each implemented <i>Secret Sharing</i> cryptographic primitive. If there is more than one implementation for the same primitive, please provide the purpose of each of them (on different rows). (e.g., Secret sharing is utilized by the TOE to securely distribute a secret among a group of users) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.1.5 EXTENDABLE-OUTPUT FUNCTIONS (XOF)

Q-ID	XOF-VQ
1	Does the TOE implement <i>XOF</i> cryptographic primitives? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>XOF</i> cryptographic primitives implemented in the TOE?
	☐ SHAKE128 ☐ SHAKE256 ☐ cSHAKE128 ☐ cSHAKE256
	☐ Other(s) - Please indicate which one(s)
3	Does the TOE implement a self-test KAT for each implemented <i>XOF</i> cryptographic primitive (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each <i>XOF</i> cryptographic primitive (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same primitive, please provide evidence of each of them (on different rows).
4	Please provide a brief description of the purpose of each implemented <i>XOF</i> cryptographic primitive. If there is more than one implementation for the

Q-ID	XOF-VQ
	same primitive, please provide the purpose of each of them (on different rows). (e.g., cSHAKE is used by the TOE as an underlying primitive to KMAC) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.2 SYMMETRIC CONSTRUCTIONS

3.2.1 SYMMETRIC ENCRYPTION (CONFIDENTIALITY ONLY)

Q-ID	SymEncryption-VQ
1	Does the TOE implement <i>Symmetric Encryption (Confidentiality Only)</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Symmetric Encryption (Confidentiality Only)</i> cryptographic constructions implemented in the TOE?
	☐ No-Padding
	☐ AES-CBC-128 ☐ PKCS#7
	☐ ISO 7816
	☐ No-Padding
	☐ AES-CBC-192 ☐ PKCS#7
	☐ ISO 7816
	☐ No-Padding
	☐ AES-CBC -256 ☐ PKCS#7
	☐ ISO 7816
	☐ AES-CBC-CS-128 ☐ AES-CBC-CS-192 ☐ AES-CBC-CS-256
	☐ AES-CFB-128 ☐ AES-CFB-192 ☐ AES-CFB-256
☐ AES-CTR-128 ☐ AES-CTR -192 ☐ AES-CTR -256	
☐ AES-OFB-128 ☐ AES-OFB-192 ☐ AES-OFB-256	
☐ Other(s) - Please indicate which one(s)	
3	Please indicate whether the implemented symmetric constructions are employed in conjunction with a cryptographic mechanism to provide data integrity and authentication. Note: Justify the answer either in the affirmative or negative case.

Q-ID	SymEncryption-VQ
4	Does the TOE implement a self-test KAT for each implemented <i>Symmetric Encryption (Confidentiality Only)</i> cryptographic construction (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each symmetric encryption cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
5	Does the TOE implement a self-test KAT for each implemented key length (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key length (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
6	Does the TOE implement a self-test KAT for each implemented direction (ciphering and deciphering) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for ciphering and deciphering operations (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
7	Please provide a brief description of the purpose of each implemented <i>Symmetric Encryption</i> cryptographic construction with its associated key length. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The AES-CBC-256 No-Padding construction is used to encrypt the data received from the input interface) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

Q-ID	SymEncryption-VQ

3.2.2 DISK ENCRYPTION

Q-ID	DiskEncryption-VQ
1	Does the TOE implement <i>Disk Encryption</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Disk Encryption</i> cryptographic constructions implemented in the TOE?
	☐ AES-XTS-256 ☐ AES-XTS-512
	☐ Other(s) - Please indicate which one(s)
3	Are XTS keys (key1 and key2) generated and/or established independently according to the rules for component symmetric keys from [SP800-133] rev2, section 6.3? In addition, does the TOE check explicitly that key1 ≠ key2 before using the keys in the XTS algorithm to process data with them? Note: In affirmative case, please, describe how both requirements are implemented and provide evidence of the implementation for each disk encryption construction (e.g., a pointer to the source code, a piece of source code, etc.).
4	Does the TOE implement a self-test KAT for each implemented <i>Disk Encryption</i> cryptographic construction (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each disk encryption construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
5	Does the TOE implement a self-test KAT for each implemented key length (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key length (e.g., a pointer to the source

Q-ID	DiskEncryption-VQ
	code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
6	Does the TOE implement a self-test KAT for each implemented direction (ciphering and deciphering) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for ciphering and deciphering operations (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
7	Please provide a brief description of the purpose of each implemented <i>Disk Encryption</i> cryptographic construction with its associated key length. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The AES-XTS-256 mode is used to perform disk encryption) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.2.3 INTEGRITY MODES: MESSAGE AUTHENTICATION CODE

Q-ID	MAC-VQ
1	Does the TOE implement <i>Message Authentication Code</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered. ☐ Yes ☐ No
2	What are the <i>Message Authentication Code</i> cryptographic constructions implemented in the TOE? ☐ AES-CMAC-128 ☐ AES-CMAC-192 ☐ AES-CMAC-256 ☐ AES-CBC-MAC-128 ☐ AES-CBC-MAC-192 ☐ AES-CBC-MAC-256

Q-ID	MAC-VQ
	☐ HMAC-SHA-1 ☐ HMAC-SHA2-256 ☐ HMAC-SHA2-384 ☐ HMAC-SHA2-512 ☐ HMAC-SHA2-512/256 ☐ HMAC-SHA3-256 ☐ HMAC-SHA3-384 ☐ HMAC-SHA3-512 ☐ AES-GMAC-128 ☐ AES-GMAC-192 ☐ AES-GMAC-256 ☐ KMAC-128 ☐ KMAC-256 ☐ Poly1305 ☐ Other(s) - Please indicate which one(s)
3	In the case of implementing the <i>HMAC</i> cryptographic construction, please provide the size of the HMAC keys used. Note: Please, provide evidence for each HMAC implementation (e.g., a pointer to the source code, a piece of source code, etc.).
4	In the case of implementing the <i>CMAC</i>, <i>CBC-MAC</i>, or <i>HMAC</i> cryptographic construction, does the TOE implement MAC truncation? What is the output bit length of the truncation? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each message authentication code cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). Note: In the case of performing MAC truncation to 64 bits, please also provide evidence that the maximal number of MAC verifications performed for a given key over its lifetime can be bounded by 2^{20}.
5	In the case of implementing the <i>GMAC</i> cryptographic construction, does the TOE comply with the following requirements related to IV management? The IV is managed within the security perimeter of the authenticated encryption process. The IV length is equal to 96 bits. The IV is constructed according to the deterministic IV construction method [SP800-38D] section 8.2.1. Note: In affirmative case, please, describe how each requirement is implemented and provide evidence of the implementation for each GMAC cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.).

Q-ID	MAC-VQ
6	In the case of implementing the <i>GMAC</i> cryptographic construction, what is the output bit length of the MAC? Note: Please, provide evidence for each GMAC implementation (e.g., a pointer to the source code, a piece of source code, etc.).
7	In the case of implementing the <i>KMAC</i> cryptographic construction, please provide the following information: a. The size of the KMAC keys used. b. Is KMAC used in XOF mode? c. What is the output bit length of the MAC? d. Does the bit length of the MAC output correspond to the direct output of the algorithm or truncation is applied? Note: Please, provide evidence for each KMAC implementation (e.g., a pointer to the source code, a piece of source code, etc.).
8	Does the TOE implement a self-test KAT for each implemented <i>Message Authentication Code</i> cryptographic construction, including tests with correct and incorrect MAC values (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each message authentication code cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
9	Does the TOE implement a self-test KAT for each underlying mechanism (hash primitive, block cipher primitive, etc.)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).

Q-ID	MAC-VQ
10	Please provide a brief description of the purpose of each implemented <i>Message Authentication Code</i> cryptographic construction with its associated key length, underlying mechanism, etc. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The HMAC-SHA-256 algorithm is used to check the integrity and authenticity of a firmware update before installing it) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.2.4 SYMMETRIC ENTITY AUTHENTICATION SCHEMES

Q-ID	EntityAuth-VQ
1	Does the TOE implement <i>Symmetric Entity Authentication</i> (challenge-response) cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the length ranges used in the <i>Symmetric Entity Authentication</i> cryptographic constructions implemented in the TOE?
	☐ $\text{len} \geq 125$ ☐ $96 \leq \text{len} < 125$ ☐ Other(s) - Please indicate which one(s)
3	Does the TOE implement a self-test KAT for each implemented <i>Symmetric Entity Authentication</i> cryptographic construction (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each symmetric entity authentication cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).

Q-ID	EntityAuth-VQ
4	Does the TOE implement a self-test KAT for each underlying mechanism (symmetric encryption construction, MAC construction (including correct and incorrect MAC values), etc.)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
5	Please provide a brief description of the purpose of each implemented <i>Symmetric Entity Authentication</i> cryptographic construction. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., A challenge-response authentication method, based on a zero-knowledge proof, is used as the initial step of communication) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.2.5 AUTHENTICATED ENCRYPTION (AE)

Q-ID	AuthEncryption-VQ
1	Does the TOE implement <i>Authenticated Encryption</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Authenticated Encryption</i> cryptographic constructions implemented in the TOE?
	☐ Encrypt-then-MAC - Please indicate the encryption and MAC mechanism
	☐ AES-CCM-128 ☐ AES-CCM-192 ☐ AES-CCM-256
	☐ AES-GCM-128 ☐ AES-GCM-192 ☐ AES-GCM-256
	☐ AES-EAX-128 ☐ AES-EAX-192 ☐ AES-EAX-256
	☐ ChaCha20-Poly1305

Q-ID	AuthEncryption-VQ
	☐ Other(s) - Please indicate which one(s)
3	In the case of implementing the <i>GCM</i> cryptographic construction, does the TOE comply with the following requirements related to IV management? a. The IV is managed within the security perimeter of the authenticated encryption process. b. The IV length is equal to 96 bits. c. The IV is constructed according to the deterministic IV construction method [SP800-38D] section 8.2.1. Note: In affirmative case, please, describe how each is implemented and provide evidence of the implementation for each GCM cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.).
4	In the case of implementing any authenticated encryption cryptographic construction, what is the output bit length of the MAC (or tag)? Note: Please, provide evidence for each authenticated encryption construction and implementation (e.g., a pointer to the source code, a piece of source code, etc.).
5	In the case of implementing the <i>ChaCha20-Poly1305</i> cryptographic construction, is the IV managed within the security perimeter of the authenticated encryption process? Note: In affirmative case, please, describe how each is implemented and provide evidence of the implementation for each ChaCha20-Poly1305 cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.).
6	Does the TOE implement a self-test KAT for each implemented <i>Authenticated Encryption</i> cryptographic construction (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each authenticated encryption cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).

Q-ID	AuthEncryption-VQ
7	Does the TOE implement a self-test KAT for each implemented key length (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key length (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
8	Does the TOE implement a self-test KAT for each implemented direction (ciphering and deciphering) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for ciphering and deciphering operations (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
9	Does the TOE implement a self-test KAT for each underlying mechanism (symmetric primitive, MAC construction (including correct and incorrect MAC values), etc.)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
10	Please provide a brief description of the purpose of each implemented <i>Authenticated Encryption</i> cryptographic construction with its associated key length. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The data sent through the communication channel is encrypted with AES-GCM-256, providing confidentiality, integrity, and authentication to the data) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.2.6 KEY PROTECTION

Q-ID	KeyProtection-VQ
1	Does the TOE implement <i>Key Protection</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Key Protection</i> cryptographic constructions implemented in the TOE?
	☐ AES-SIV-256 ☐ AES-SIV-384 ☐ AES-SIV-512
	☐ AES-KW-128 ☐ AES-KW-192 ☐ AES-KW-256
	☐ AES-KWP-128 ☐ AES-KWP-192 ☐ AES-KWP-256
	☐ Authenticated Encryption Construction(s) - Please indicate which one(s)
	☐ Other(s) - Please indicate which one(s)
3	In the case of implementing an Authenticated Encryption construction for <i>Key Protection</i> functionality, does the TOE implement all the self-test KATs for each implemented authenticated encryption construction, key length, direction, and underlying mechanisms? Note 1: If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows). Note 2: Please, complete Q-6, Q-7, Q-8, and Q-9 of the AuthEncryption-VQ table for each authenticated encryption construction implemented for Key Protection functionality.
4	Does the TOE implement a self-test KAT for each implemented <i>Key Protection</i> cryptographic construction (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key protection cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).

Q-ID	KeyProtection-VQ
5	Does the TOE implement a self-test KAT for each implemented key length (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key length (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
6	Does the TOE implement a self-test KAT for each implemented direction (ciphering and deciphering) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for ciphering and deciphering operations (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
7	Does the TOE implement a self-test KAT for each underlying mechanism (symmetric primitive, MAC construction (including correct and incorrect MAC values), etc.)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
8	Please provide a brief description of the purpose of each implemented <i>Key Protection</i> cryptographic construction with its associated key length. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The SSPs are entered into the TOE encrypted with AES-KWP-256) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.2.7 KEY DERIVATION FUNCTIONS (KDF)

Q-ID	KeyDerivation-VQ
1	Does the TOE implement <i>Key Derivation</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Key Derivation</i> cryptographic constructions implemented in the TOE?
	☐ NIST SP800-56A/B/C One-Step KDF
	☐ NIST SP800-56A/B/C Two-Step KDF
	☐ NIST SP800-108 KDF in Counter Mode
	☐ NIST SP800-108 KDF in Feedback Mode
	☐ NIST SP800-108 KDF in Double-Pipeline Mode
	☐ NIST SP800-108 KDF using KMAC
	☐ ANSI-X9.63 KDF
	☐ PBKDF2
3	☐ HKDF (RFC 5869)
	☐ Other(s) - Please indicate which one(s)
	In the case of implementing the <i>NIST SP800-56A/B/C One-Step KDF</i> cryptographic construction, please indicate its parameterization.
	☐ SHA2-256 ☐ SHA2-384 ☐ SHA2-512
	☐ SHA2-512/256 ☐ SHA3-256 ☐ SHA3-384
	☐ SHA3-512
	☐ HMAC SHA1 ☐ HMAC SHA2-256 ☐ HMAC SHA2-384
	☐ HMAC SHA2-512 ☐ HMAC SHA2-512/256 ☐ HMAC SHA3-256
	☐ HMAC SHA3-384 ☐ HMAC SHA3-512
☐ KMAC-128 ☐ KMAC-256	

Q-ID	KeyDerivation-VQ	
	☐ Other(s) - Please indicate which one(s)	
4	In the case of implementing the <i>NIST SP800-56A/B/C Two-Step KDF</i> cryptographic construction, please indicate its parameterization.	
	Extraction Step	
	☐ HMAC SHA1 ☐ HMAC SHA2-256 ☐ HMAC SHA2-384 ☐ HMAC SHA2-512 ☐ HMAC SHA2-512/256 ☐ HMAC SHA3-256 ☐ HMAC SHA3-384 ☐ HMAC SHA3-512	
	☐ AES-128-CMAC ☐ AES-192-CMAC ☐ AES-256-CMAC	
	☐ Other(s) - Please indicate which one(s)	
	Expansion Step	
	☐ NIST SP800-108 KDF Please indicate which one(s) and the complete parameterization. It is necessary to complete the associated questions (Q-5 to Q-8) for the implemented NIST SP800-108 KDFs.	
	☐ Other(s) - Please indicate which one(s)	
	5	In the case of implementing the <i>NIST SP800-108 KDF in Counter Mode</i> cryptographic construction, please indicate its parameterization.
		Underlying Mechanisms
☐ HMAC SHA1 ☐ HMAC SHA2-256 ☐ HMAC SHA2-384 ☐ HMAC SHA2-512 ☐ HMAC SHA2-512/256 ☐ HMAC SHA3-256 ☐ HMAC SHA3-384 ☐ HMAC SHA3-512		
☐ AES-128-CMAC ☐ AES-192-CMAC ☐ AES-256-CMAC		
☐ KMAC-128 ☐ KMAC-256		

Q-ID	KeyDerivation-VQ
	☐ Other(s) - Please indicate which one(s)
	Supported Parameterization
	Counter i : XXX bits ☐ Big-endian ☐ Little-endian
	Requested length L : XXX bits ☐ Big-endian ☐ Little-endian
	Label: ☐ Yes ☐ No
	Context: ☐ Yes ☐ No
	All-zero octet (z): ☐ Yes ☐ No
	The pattern for the input to the underlying PRF: (For example, $i \text{Label} 0x00 \text{Context} L$)
6	In the case of implementing the <i>NIST SP800-108 KDF in Feedback Mode</i> cryptographic construction, please indicate its parameterization.
	Underlying Mechanisms
	☐ HMAC SHA1 ☐ HMAC SHA2-256
	☐ HMAC SHA2-384 ☐ HMAC SHA2-512
	☐ HMAC SHA2-512/256 ☐ HMAC SHA3-256
	☐ HMAC SHA3-384 ☐ HMAC SHA3-512
	☐ AES-128-CMAC ☐ AES-192-CMAC
	☐ AES-256-CMAC
	☐ KMAC-128 ☐ KMAC-256
	☐ Other(s) - Please indicate which one(s)
	Supported Parameterization
	Counter i : XXX bits ☐ Big-endian ☐ Little-endian
	Requested length L : XXX bits ☐ Big-endian ☐ Little-endian
	IV: ☐ Yes ☐ No
Label: ☐ Yes ☐ No	

Q-ID	KeyDerivation-VQ
	Context: ☐ Yes ☐ No
	All-zero octet (z): ☐ Yes ☐ No
	The pattern for the input to the underlying PRF: (For example, IV i Label 0x00 Context L)
7	In the case of implementing the <i>NIST SP800-108 KDF in Double-Pipeline Mode</i> cryptographic construction, please indicate its parameterization.
	Underlying Mechanisms
	☐ HMAC SHA1 ☐ HMAC SHA2-256
	☐ HMAC SHA2-384 ☐ HMAC SHA2-512
	☐ HMAC SHA2-512/256 ☐ HMAC SHA3-256
	☐ HMAC SHA3-384 ☐ HMAC SHA3-512
	☐ AES-128-CMAC ☐ AES-192-CMAC
	☐ AES-256-CMAC
	☐ KMAC-128 ☐ KMAC-256
	☐ Other(s) - Please indicate which one(s)
	Supported Parameterization
	Counter <i>i</i> : XXX bits ☐ Big-endian ☐ Little-endian
	Requested length <i>L</i> : XXX bits ☐ Big-endian ☐ Little-endian
	Label: ☐ Yes ☐ No
	Context: ☐ Yes ☐ No
All-zero octet (z): ☐ Yes ☐ No	
The pattern for the input to the underlying PRF: (For example, i Label 0x00 Context L)	
8	In the case of implementing the <i>NIST SP800-108 KDF in KMAC mode</i> cryptographic construction, please indicate its parameterization.
	☐ KMAC-128 ☐ KMAC-256

Q-ID	KeyDerivation-VQ
	☐ Other(s) - Please indicate which one(s)
9	In the case of implementing the <i>ANSI-X9.63 KDF</i> cryptographic construction, please indicate its parameterization.
	☐ SHA2-256 ☐ SHA2-384
	☐ SHA2-512 ☐ SHA2-512/256
	☐ SHA3-256 ☐ SHA3-384
	☐ SHA3-512
	☐ Other(s) - Please indicate which one(s)
10	In the case of implementing the <i>PBKDF2</i> cryptographic construction, please indicate its parameterization.
	☐ HMAC-SHA-1 ☐ HMAC-SHA2-256 ☐ HMAC-SHA2-384
	☐ HMAC-SHA2-512 ☐ HMAC-SHA2-512/256 ☐ HMAC-SHA3-256
	☐ HMAC-SHA3-384 ☐ HMAC-SHA3-512
	☐ Other(s) - Please indicate which one(s)
11	In the case of implementing the <i>PBKDF2</i> cryptographic construction, what is the bit length of the HMAC key?
12	In the case of implementing the <i>HKDF (RFC 5869)</i> cryptographic construction, please indicate its parameterization.
	☐ HMAC-SHA-1 ☐ HMAC-SHA2-256 ☐ HMAC-SHA2-384
	☐ HMAC-SHA2-512 ☐ HMAC-SHA2-512/256 ☐ HMAC-SHA3-256
	☐ HMAC-SHA3-384 ☐ HMAC-SHA3-512
	☐ Other(s) - Please indicate which one(s)
13	Does the TOE implement a self-test KAT for each <i>Key Derivation</i> cryptographic construction implemented in the TOE? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key derivation function (e.g., a pointer

Q-ID	KeyDerivation-VQ
	to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
14	Does the TOE implement a self-test KAT for each underlying mechanism (hash primitive, MAC (including correct and incorrect MAC values), etc.)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
15	Please provide a brief description of the purpose of each implemented <i>Key Derivation</i> cryptographic construction with its associated underlying mechanism, length of the derived key, etc. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The PBKDF2-HMAC-SHA-256 derives a new key of 256 bits using the user password) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.2.8 PASSWORD PROTECTION/HASHING MECHANISMS

Q-ID	PasswordMechanisms-VQ
1	Does the TOE implement <i>Password Protection/Hashing</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered. ☐ Yes ☐ No
2	What are the <i>Password Protection/Hashing</i> cryptographic constructions implemented in the TOE? ☐ Argon2id

Q-ID	PasswordMechanisms-VQ
	☐ PBKDF2
	☐ SCRYPT
	☐ Other(s) - Please indicate which one(s)
3	In the case of implementing the <i>PBKDF2</i> cryptographic construction, please indicate its parameterization.
	☐ HMAC-SHA-1 ☐ HMAC-SHA2-256 ☐ HMAC-SHA2-384
	☐ HMAC-SHA2-512 ☐ HMAC-SHA2-512/256 ☐ HMAC-SHA3-256
	☐ HMAC-SHA3-384 ☐ HMAC-SHA3-512
	☐ Other(s) - Please indicate which one(s)
4	In the case of implementing the <i>SCRYPT</i> cryptographic construction, please indicate its parameterization.
	☐ PBKDF2 HMAC SHA2-256
	☐ Other(s) - Please indicate which one(s)
5	In the case of implementing the <i>Argon2id</i> cryptographic construction, please indicate the following parameterization.
	Note: Please, provide evidence for each Argon2id implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	The memory size (m) is: XXX
	The number of passes (t) is: XXX passes
	The password length (P) is XXX bytes
	The degree of parallelism (p) is XXX lanes
	The salt (S) length is: XXX bits
	Is the salt unique for each password? ☐ Yes ☐ No
	The tag (T) length is: XXX bits
	The secret value (K) length is: XXX bytes
The associated data (X) length is: XXX bytes	

Q-ID	PasswordMechanisms-VQ
6	In the case of implementing the <i>PBKDF2</i> cryptographic construction, please indicate the following parameterization. Note: Please, provide evidence for each PBKDF2 implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	The number of iterations is: XXX iterations
	The salt length is: XXX bits
7	Does the TOE implement a self-test KAT for each <i>Password Protection/Hashing</i> cryptographic construction implemented in the TOE? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each password protection/ hashing cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
8	Does the TOE implement a self-test KAT for each underlying mechanism (hash primitive, MAC (including correct and incorrect MAC values), etc.)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
9	Please provide a brief description of the purpose of each implemented <i>Password Protection/Hashing</i> cryptographic with its associated underlying mechanism, length of the derived key, etc. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The TOE protects the user password using PBKDF2-HMAC-SHA-256 cryptographic construction) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.3 ASYMMETRIC ELEMENTARY PRIMITIVES

3.3.1 RSA/INTEGER FACTORIZATION

Q-ID	RSA-VQ
1	Does the TOE implement cryptographic primitives based on the <i>integer factorization problem (RSA)</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>RSA</i> cryptographic primitives implemented in the TOE?
	☐ RSA-2048 bits ☐ RSA-3072 bits ☐ RSA-4096 bits ☐ Other(s) - Please indicate which one(s)
3	Please provide a brief description of the purpose of each implemented cryptographic primitive based on the <i>integer factorization problem (RSA)</i>. If there is more than one implementation for the same primitive, please provide the purpose of each of them (on different rows). (e.g., The TOE uses a digital signature based on RSA-2048 to perform firmware authentication) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

Q-ID	RSA-IMPLEMENTATION-VQ
1	What are the <i>RSA</i> cryptographic operations implemented in the TOE? Please complete a row for each of the different <i>RSA</i> parameterizations and/or implementations. Add as many as necessary.
1-1	RSA Implementation – 1
	☐ Modular Exponentiation operations with an <i>RSA</i> Public Key
	☐ Modular Exponentiation operations with an <i>RSA</i> Public Key, where the value of <i>e</i> is equal to 65537
	☐ Modular Exponentiation operations with an <i>RSA</i> Private Key
	☐ Modular Exponentiation operations with extended parameters using recombination of the Chinese Remainder Theorem (CRT)

	☐ Other(s) - Please indicate which one(s)
1-2	RSA Implementation – 2
	☐ Modular Exponentiation operations with an RSA Public Key
	☐ Modular Exponentiation operations with an RSA Public Key, where the value of e is equal to 65537
	☐ Modular Exponentiation operations with an RSA Private Key
	☐ Modular Exponentiation operations with extended parameters using recombination of the Chinese Remainder Theorem (CRT)
	☐ Other(s) - Please indicate which one(s)
1-3	...
...	...

3.3.2 MULTIPLICATIVE DISCRETE LOGARITHM PROBLEM (FFDLOG)

Q-ID	FFDLOG-VQ
1	Does the TOE implement <i>FFDLOG</i> cryptographic primitives? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>FFDLOG</i> cryptographic primitives implemented in the TOE?
	☐ MODP-2048 bits ☐ MODP-3072 bits ☐ MODP-4096 bits
	☐ MODP-6144 bits ☐ MODP-8192 bits
	☐ FFDHE-2048 bits ☐ FFDHE-3072 bits ☐ FFDHE-4096 bits
	☐ FFDHE-6144 bits ☐ FFDHE-8192 bits
	☐ Other(s) - Please indicate which one(s)
3	Please provide a brief description of the purpose of each implemented asymmetric cryptographic primitive based on <i>FFDLOG</i>. If there is more than one implementation for the same primitive, please provide the purpose of each of them (on different rows). (e.g., The FFDHE-2048 group is used in the DH key establishment scheme of the cipher suites utilized by TLS v1.2)

Q-ID	FFDLOG-VQ
	Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.3.3 ADDITIVE DISCRETE LOGARITHM PROBLEM (ECDLOG)

Q-ID	ECDLOG-VQ
1	Does the TOE implement <i>ECDLOG</i> cryptographic primitives? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>ECDLOG</i> cryptographic primitives implemented in the TOE?
	☐ BrainpoolP256r1 ☐ BrainpoolP384r1 ☐ BrainpoolP512r1
	☐ NIST P-256 ☐ NIST P-384 ☐ NIST P-521
	☐ FRP256v1 ☐ Curve25519 ☐ Curve448
	☐ Edwards25519 ☐ Edwards448
	☐ Other(s) - Please indicate which one(s)
3	Please provide a brief description of the purpose of each implemented asymmetric cryptographic primitive based on <i>ECDLOG</i>. If there is more than one implementation for the same primitive, please provide the purpose of each of them (on different rows). (e.g., The NIST P-256 curve is used in the ECDHE key establishment scheme of the cipher suites utilized by TLS v1.2) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

Q-ID	ECDLOG-IMPLEMENTATION-VQ
1	What are the <i>Elliptic Curve</i> cryptographic operations implemented in the TOE? Please complete a row for each of the different ECDLOG parameterizations and/or implementations. Add as many as necessary. In the case of NIST P Curves, BrainpoolP Curves, and FRP256v1 Curve:

Q-ID	ECDLOG-IMPLEMENTATION-VQ
1-1	ECDLOG Implementation – 1
	☐ Key Generation Please indicate the associated operation
	☐ EC Scalar Multiplication with Fixed Basepoint $R = d \times G$ where G denotes the generator
	☐ EC Scalar Multiplication $R = d \times P$ where P is generic and G is introduced into it as the generator
	☐ Other(s) - Please indicate which one(s)
	☐ Signature Generation Please indicate the associated operation
	☐ EC Scalar Multiplication $R = d \times P$ where P is a point lying on the curve
	☐ Other(s) - Please indicate which one(s)
	☐ Signature Verification Please indicate the associated operation
	☐ MultiScalar Multiplication $R = u \times S + v \times T$ where S and T are points lying on the curve S is generic and G is introduced into it as the generator
	☐ MultiScalar Multiplication with Fixed Basepoint $R = u \times G + v \times T$ where T is a point lying on the curve and G denotes the generator
	☐ Other(s) - Please indicate which one(s)
	☐ Preshared Secret Computation Please indicate the associated operation
	☐ EC Scalar Multiplication $R = d \times P$ where P is a point lying on the curve
	☐ Other(s) - Please indicate which one(s)
1-2	...
...	...
2	What are the <i>Elliptic Curve</i> cryptographic operations implemented in the TOE?

Q-ID	ECDLOG-IMPLEMENTATION-VQ	
	Please complete a row for each of the different ECDLOG parameterizations and/or implementations. Add as many as necessary. In the case of Curve25519 and Curve448:	
2-1	ECDLOG Implementation – 1	
	☐ Key Generation Please indicate the associated operation	☐ EC Scalar Multiplication with Fixed Basepoint $R = d \times G$ where G denotes the generator
		☐ EC Scalar Multiplication $R = d \times P$ where P is generic and G is introduced into it as the generator
		☐ Other(s) - Please indicate which one(s)
	☐ Preshared Secret Computation Please indicate the associated operation	☐ EC Scalar Multiplication $R = d \times P$ where P is a point lying on the curve
		☐ Other(s) - Please indicate which one(s)
2-2	...	
2-3	...	
3	What are the <i>Elliptic Curve</i> cryptographic operations implemented in the TOE? Please complete a row for each of the different ECDLOG parameterizations and/or implementations. Add as many as necessary. In the case of Edwards25519 and Edwards448 curves:	
3-1	ECDLOG Implementation – 1	
	☐ Key Generation Please indicate the associated operation	☐ EC Scalar Multiplication with Fixed Basepoint $R = d \times G$ where G denotes the generator
		☐ EC Scalar Multiplication $R = d \times P$ where P is generic and G is introduced into it as the generator

Q-ID	ECDLOG-IMPLEMENTATION-VQ	
		☐ Other(s) - Please indicate which one(s)
	☐ Signature Generation Please indicate the associated operation	☐ EC Scalar Multiplication $R = d \times P$ where P is a point lying on the curve ☐ Other(s) - Please indicate which one(s)
	☐ Signature Verification Please indicate the associated operation	☐ MultiScalar Multiplication $R = u \times S + v \times T$ where S and T are points lying on the curve S is generic and G is introduced into it as the generator ☐ MultiScalar Multiplication with Fixed Basepoint $R = u \times G + v \times T$ where T is a point lying on the curve and G denotes the generator ☐ Other(s) - Please indicate which one(s)
3-2	...	
3-3	...	

3.4 ASYMMETRIC CONSTRUCTIONS

3.4.1 ASYMMETRIC ENCRYPTION SCHEMES

Q-ID	AsymEncryption-VQ																																
1	Does the TOE implement <i>Asymmetric Encryption</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.																																
	☐ Yes ☐ No																																
2	What are the <i>Asymmetric Encryption</i> cryptographic constructions implemented in the TOE?																																
	☐ RSA-OAEP (PKCS#1v2.1) ☐ RSA-PKCS#1v1.5 ☐ Other(s) - Please indicate which one(s)																																
3	In the case of implementing the <i>RSA-OAEP (PKCS#1v2.1)</i> cryptographic construction, please indicate its parameterization.																																
	<table border="1"> <thead> <tr> <th></th><th>Mod 2048-bit</th><th>Mod 3072-bit</th><th>Mod 4096-bit</th></tr> </thead> <tbody> <tr> <td>SHA2-256</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-384</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512/256</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-256</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-384</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-512</td><td>☐</td><td>☐</td><td>☐</td></tr> </tbody> </table>		Mod 2048-bit	Mod 3072-bit	Mod 4096-bit	SHA2-256	☐	☐	☐	SHA2-384	☐	☐	☐	SHA2-512	☐	☐	☐	SHA2-512/256	☐	☐	☐	SHA3-256	☐	☐	☐	SHA3-384	☐	☐	☐	SHA3-512	☐	☐	☐
		Mod 2048-bit	Mod 3072-bit	Mod 4096-bit																													
	SHA2-256	☐	☐	☐																													
	SHA2-384	☐	☐	☐																													
	SHA2-512	☐	☐	☐																													
	SHA2-512/256	☐	☐	☐																													
	SHA3-256	☐	☐	☐																													
SHA3-384	☐	☐	☐																														
SHA3-512	☐	☐	☐																														
☐ Other(s) - Please indicate which one(s)																																	
4	In the case of implementing the <i>RSA Asymmetric Encryption</i> cryptographic construction, please indicate the agreed deterministic random number generator used to generate the randomized padding. Note: Please, provide evidence for each RSA asymmetric encryption implementation (e.g., a pointer to the source code, a piece of source code, etc.). If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).																																

Q-ID	AsymEncryption-VQ
5	In the case of implementing the <i>RSA-PKCS#1v1.5</i> cryptographic construction, please indicate its parameterization.
	☐ RSA-2048 bits ☐ RSA-3072 bits ☐ RSA-4096 bits ☐ Other(s) - Please indicate which one(s)
6	Does the TOE implement a self-test KAT for each implemented <i>Asymmetric Encryption</i> cryptographic construction (even if they are not in use), including correct and incorrect encryption operations? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each asymmetric encryption cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
7	Does the TOE implement a self-test KAT for each underlying mechanism (hash primitive)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
8	Please provide a brief description of the purpose of each implemented <i>Asymmetric Encryption</i> cryptographic construction with its associated modulus, hash primitive, etc. If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., User data are entered into the TOE encrypted with RSA-OAEP with modulus 2048 bits and SHA-256) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.4.2 DIGITAL SIGNATURE

Q-ID	DigitalSignature-VQ																																
1	Does the TOE implement <i>Digital Signature</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.																																
	☐ Yes ☐ No																																
2	What are the <i>Digital Signature</i> cryptographic constructions implemented in the TOE?																																
	☐ RSA-PSS (PKCS#1v2.1) ☐ RSA-PKCS#1v1.5																																
	☐ DSA ☐ KCDSA ☐ Schnorr (SDSA)																																
	☐ EC-DSA ☐ EC-KCDSA ☐ EC-GDSA																																
	☐ EC-Schnorr (EC-SDSA) ☐ EC-Schnorr (EC-FSDSA) ☐ EdDSA																																
	☐ XMSS																																
	☐ ML-DSA ☐ Falcon ☐ SLH-DSA																																
3	In the case of implementing the <i>RSA-PSS (PKCS#1v2.1)</i> cryptographic construction, please indicate its parameterization.																																
	<table border="1"> <thead> <tr> <th></th><th>Mod 2048-bit</th><th>Mod 3072-bit</th><th>Mod 4096-bit</th></tr> </thead> <tbody> <tr> <td>SHA2-256</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-384</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512/256</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-256</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-384</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-512</td><td>☐</td><td>☐</td><td>☐</td></tr> </tbody> </table>		Mod 2048-bit	Mod 3072-bit	Mod 4096-bit	SHA2-256	☐	☐	☐	SHA2-384	☐	☐	☐	SHA2-512	☐	☐	☐	SHA2-512/256	☐	☐	☐	SHA3-256	☐	☐	☐	SHA3-384	☐	☐	☐	SHA3-512	☐	☐	☐
		Mod 2048-bit	Mod 3072-bit	Mod 4096-bit																													
	SHA2-256	☐	☐	☐																													
	SHA2-384	☐	☐	☐																													
	SHA2-512	☐	☐	☐																													
	SHA2-512/256	☐	☐	☐																													
	SHA3-256	☐	☐	☐																													
SHA3-384	☐	☐	☐																														
SHA3-512	☐	☐	☐																														
☐ Other(s) - Please indicate which one(s)																																	
4	In the case of implementing the <i>RSA-PKCS#1v1.5</i> cryptographic construction, please indicate its parameterization.																																

Q-ID	DigitalSignature-VQ					
		Mod 2048-bit		Mod 3072-bit		Mod 4096-bit
	SHA2-256	☐		☐		☐
	SHA2-384	☐		☐		☐
	SHA2-512	☐		☐		☐
	SHA2-512/256	☐		☐		☐
	☐ Other(s) - Please indicate which one(s)					
5	In the case of implementing the <i>DSA</i> cryptographic construction, please indicate its parameterization.					
		MODP 2048-bit		MODP 3072-bit		MODP 4096-bit
		MODP 6144-bit		MODP 8192-bit		
	SHA2-256	☐	☐	☐	☐	☐
	SHA2-384	☐	☐	☐	☐	☐
	SHA2-512	☐	☐	☐	☐	☐
	SHA2-512/256	☐	☐	☐	☐	☐
	SHA3-256	☐	☐	☐	☐	☐
	SHA3-384	☐	☐	☐	☐	☐
	SHA3-512	☐	☐	☐	☐	☐
	☐ Other(s) - Please indicate which one(s)					
6	In the case of implementing the <i>KCDSA</i> cryptographic construction, please indicate its parameterization.					
		MODP 2048-bit		MODP 3072-bit		MODP 4096-bit
		MODP 6144-bit		MODP 8192-bit		
	SHA2-256	☐	☐	☐	☐	☐
	SHA2-384	☐	☐	☐	☐	☐
	SHA2-512	☐	☐	☐	☐	☐
	SHA2-512/256	☐	☐	☐	☐	☐
	SHA3-256	☐	☐	☐	☐	☐
SHA3-384	☐	☐	☐	☐	☐	

Q-ID	DigitalSignature-VQ																																																																
	SHA3-512 ☐ ☐ ☐ ☐ ☐ ☐ Other(s) - Please indicate which one(s)																																																																
7	In the case of implementing the <i>SDSA</i> cryptographic construction, please indicate its parameterization.																																																																
	<table border="1"> <thead> <tr> <th></th><th>MODP 2048-bit</th><th>MODP 3072-bit</th><th>MODP 4096-bit</th><th>MODP 6144-bit</th><th>MODP 8192-bit</th></tr> </thead> <tbody> <tr> <td>SHA2-256</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-384</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512/256</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-256</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-384</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-512</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> </tbody> </table>		MODP 2048-bit	MODP 3072-bit	MODP 4096-bit	MODP 6144-bit	MODP 8192-bit	SHA2-256	☐	☐	☐	☐	☐	SHA2-384	☐	☐	☐	☐	☐	SHA2-512	☐	☐	☐	☐	☐	SHA2-512/256	☐	☐	☐	☐	☐	SHA3-256	☐	☐	☐	☐	☐	SHA3-384	☐	☐	☐	☐	☐	SHA3-512	☐	☐	☐	☐	☐																
		MODP 2048-bit	MODP 3072-bit	MODP 4096-bit	MODP 6144-bit	MODP 8192-bit																																																											
	SHA2-256	☐	☐	☐	☐	☐																																																											
	SHA2-384	☐	☐	☐	☐	☐																																																											
	SHA2-512	☐	☐	☐	☐	☐																																																											
	SHA2-512/256	☐	☐	☐	☐	☐																																																											
	SHA3-256	☐	☐	☐	☐	☐																																																											
SHA3-384	☐	☐	☐	☐	☐																																																												
SHA3-512	☐	☐	☐	☐	☐																																																												
☐ Other(s) - Please indicate which one(s)																																																																	
8	In the case of implementing the <i>EC-DSA</i> cryptographic construction, please indicate its parameterization.																																																																
	<table border="1"> <thead> <tr> <th></th><th>P-256</th><th>P-384</th><th>P-521</th><th>Brainpool P256r1</th><th>Brainpool P384r1</th><th>Brainpool P512r1</th><th>FRP256v1</th></tr> </thead> <tbody> <tr> <td>SHA2-256</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-384</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA2-512 /256</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-256</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-384</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> <tr> <td>SHA3-512</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td><td>☐</td></tr> </tbody> </table>		P-256	P-384	P-521	Brainpool P256r1	Brainpool P384r1	Brainpool P512r1	FRP256v1	SHA2-256	☐	☐	☐	☐	☐	☐	☐	SHA2-384	☐	☐	☐	☐	☐	☐	☐	SHA2-512	☐	☐	☐	☐	☐	☐	☐	SHA2-512 /256	☐	☐	☐	☐	☐	☐	☐	SHA3-256	☐	☐	☐	☐	☐	☐	☐	SHA3-384	☐	☐	☐	☐	☐	☐	☐	SHA3-512	☐	☐	☐	☐	☐	☐	☐
		P-256	P-384	P-521	Brainpool P256r1	Brainpool P384r1	Brainpool P512r1	FRP256v1																																																									
	SHA2-256	☐	☐	☐	☐	☐	☐	☐																																																									
	SHA2-384	☐	☐	☐	☐	☐	☐	☐																																																									
	SHA2-512	☐	☐	☐	☐	☐	☐	☐																																																									
	SHA2-512 /256	☐	☐	☐	☐	☐	☐	☐																																																									
	SHA3-256	☐	☐	☐	☐	☐	☐	☐																																																									
SHA3-384	☐	☐	☐	☐	☐	☐	☐																																																										
SHA3-512	☐	☐	☐	☐	☐	☐	☐																																																										
☐ Other(s) - Please indicate which one(s)																																																																	
9	In the case of implementing the <i>EC-KCDSA</i> cryptographic construction, please indicate its parameterization.																																																																

Q-ID	DigitalSignature-VQ							
		P-256	P-384	P-521	Brainpool P256r1	Brainpool P384r1	Brainpool P512r1	FRP256v1
	SHA2-256	☐	☐	☐	☐	☐	☐	☐
	SHA2-384	☐	☐	☐	☐	☐	☐	☐
	SHA2-512	☐	☐	☐	☐	☐	☐	☐
	SHA2-512 /256	☐	☐	☐	☐	☐	☐	☐
	SHA3-256	☐	☐	☐	☐	☐	☐	☐
	SHA3-384	☐	☐	☐	☐	☐	☐	☐
	SHA3-512	☐	☐	☐	☐	☐	☐	☐
☐ Other(s) - Please indicate which one(s)								
10	In the case of implementing the <i>EC-GDSA</i> cryptographic construction, please indicate its parameterization.							
		P-256	P-384	P-521	Brainpool P256r1	Brainpool P384r1	Brainpool P512r1	FRP256v1
	SHA2-256	☐	☐	☐	☐	☐	☐	☐
	SHA2-384	☐	☐	☐	☐	☐	☐	☐
	SHA2-512	☐	☐	☐	☐	☐	☐	☐
	SHA2-512 /256	☐	☐	☐	☐	☐	☐	☐
	SHA3-256	☐	☐	☐	☐	☐	☐	☐
	SHA3-384	☐	☐	☐	☐	☐	☐	☐
	SHA3-512	☐	☐	☐	☐	☐	☐	☐
☐ Other(s) - Please indicate which one(s)								
11	In the case of implementing the <i>EC-SDSA</i> cryptographic construction, please indicate its parameterization.							
		P-256	P-384	P-521	Brainpool P256r1	Brainpool P384r1	Brainpool P512r1	FRP256v1
	SHA2-256	☐	☐	☐	☐	☐	☐	☐
	SHA2-384	☐	☐	☐	☐	☐	☐	☐

Q-ID	DigitalSignature-VQ
	SHA2-512 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA2-512 /256 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA3-256 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA3-384 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA3-512 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	☐ Other(s) - Please indicate which one(s)
12	In the case of implementing the <i>EC-FSDSA</i> cryptographic construction, please indicate its parameterization.
	P-256 P-384 P-521 Brainpool P256r1 Brainpool P384r1 Brainpool P512r1 FRP256v1
	SHA2-256 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA2-384 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA2-512 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA2-512 /256 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA3-256 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA3-384 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	SHA3-512 ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐
	☐ Other(s) - Please indicate which one(s)
13	In the case of implementing the <i>EdDSA</i> cryptographic construction, please indicate its parameterization.
	Ed25519 with SHA2-512 Ed448 with SHAKE256
	PureEdDSA ☐ ☐
	Pre-Hash ☐ ☐
☐ Other(s) - Please indicate which one(s)	
14	In the case of implementing any <i>digital signature</i> cryptographic mechanism based on DSA or its elliptic curve variants, what are the random uniform schemes

Q-ID	DigitalSignature-VQ
	used by the TOE for the generation of the random value k? (non-applicable for EdDSA schemes) Note: For each scheme employed, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ Testing Technique – Algorithm 1, according to [CCN-STIC-221] guide
	☐ Extra Random Technique – Algorithm 2, according to [CCN-STIC-221] guide
	☐ Other(s) - Please indicate which one(s)
	Do the implemented random uniform schemes apply to the output of an agreed deterministic random number generator?
	☐ Yes ☐ No
15	In the case of implementing the <i>XMSS</i> cryptographic construction, please indicate its parameterization.
	☐ SHA2-256 ☐ SHA2-512
	☐ SHAKE128 ☐ SHAKE256
	☐ Other(s) - Please indicate which one(s)
16	In the case of implementing the <i>ML-DSA</i> cryptographic construction, please indicate its parameterization.
	☐ Supported Configuration(s) - Please indicate which one(s)
17	In the case of implementing the <i>Falcon</i> cryptographic construction, please indicate its parameterization.
	☐ Supported Configuration(s) - Please indicate which one(s)
18	In the case of implementing the <i>SLH-DSA</i> cryptographic construction, please indicate its parameterization.
	☐ Supported Configuration(s) - Please indicate which one(s)
19	Does the TOE implement a self-test KAT for each implemented <i>Digital Signature</i> cryptographic construction (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each digital signature cryptographic

Q-ID	DigitalSignature-VQ
	construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
20	Does the TOE implement a self-test KAT for each implemented <i>Digital Signature</i> configuration (RSA modulus and scheme, finite field group, and elliptic curve) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each configuration (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
21	Does the TOE implement a self-test KAT for each implemented <i>Digital Signature</i> operation (generation and verification) (even if they are not in use), including correct and incorrect signature values for the verification operation? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each digital signature generation and verification operation (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
22	Does the TOE implement a self-test KAT for each underlying mechanism (hash primitive, XOF primitive, etc.)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
23	Please provide a brief description of the purpose of each implemented <i>Digital Signature</i> cryptographic construction with its associated RSA modulus and scheme, finite field group, elliptic curve, hash primitive, etc. If there is more than

Q-ID	DigitalSignature-VQ
	one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., Data sent through the communication channel is signed using ECDSA with P-256 curve with SHA-256 hash primitive) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.4.3 KEY ESTABLISHMENT

Q-ID	KeyEstablishment-VQ
1	Does the TOE implement <i>Key Establishment</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>Key Establishment</i> cryptographic constructions implemented in the TOE?
	☐ Diffie-Hellman (DH) ☐ DLIES-KEM
	☐ Elliptic Curve Diffie-Hellman (ECDH) ☐ ECIES-KEM
	☐ ML-KEM ☐ FrodoKEM
	☐ Other(s) - Please indicate which one(s)
3	In the case of implementing the <i>Diffie-Hellman (DH)</i> cryptographic construction, please indicate its parameterization.
	☐ Non-Ephemeral ☐ Ephemeral In case both apply, indicate which parameterization applies to each.
	☐ MODP-2048 bits ☐ MODP-3072 bits ☐ MODP-4096 bits
	☐ MODP-6144 bits ☐ MODP-8192 bits
	☐ FFDHE-2048 bits ☐ FFDHE-3072 bits ☐ FFDHE-4096 bits ☐ FFDHE-6144 bits ☐ FFDHE-8192 bits

Q-ID	KeyEstablishment-VQ
	☐ Key Derivation Function - Please indicate which one(s) Please, complete the information according to those provided in Section 3.2.7 "Key Derivation Functions".
	☐ Other(s) Configuration(s) - Please indicate which one(s)
4	In the case of implementing the <i>DLIES-KEM</i> cryptographic construction, please indicate its parameterization.
	☐ Non-Ephemeral ☐ Ephemeral In case both apply, indicate which parameterization applies to each.
	☐ MODP-3072 bits ☐ MODP-4096 bits ☐ FFDHE-3072 bits ☐ FFDHE-4096 bits
	Key Derivation Function: ☐ HKDF with HMAC-SHA2-256
	☐ Other(s) Configuration(s) - Please indicate which one(s)
5	In the case of implementing the <i>Elliptic Curve Diffie-Hellman (EC-DH)</i> cryptographic construction, please indicate its parameterization.
	☐ Non-Ephemeral ☐ Ephemeral In case both apply, indicate which parameterization applies to each.
	☐ BrainpoolP256r1 ☐ BrainpoolP384r1 ☐ BrainpoolP512r1 ☐ NIST P-256 ☐ NIST P-384 ☐ NIST P-521 ☐ FRP256v1 ☐ Curve25519 (x25519) ☐ Curve448 (x448)
	☐ Key Derivation Function - Please indicate which one(s) Please, complete the information according to those provided in Section 3.2.7 "Key Derivation Functions".
	☐ Other(s) Configuration(s) - Please indicate which one(s)
6	In the case of implementing the <i>ECIES-KEM</i> cryptographic construction, please indicate its parameterization.
	☐ Non-Ephemeral ☐ Ephemeral In case both apply, indicate which parameterization applies to each.

Q-ID	KeyEstablishment-VQ
	☐ BrainpoolP256r1 ☐ BrainpoolP384r1 ☐ BrainpoolP512r1 ☐ NIST P-256 ☐ NIST P-384 ☐ NIST P-521 Key Derivation Function: ☐ HKDF with HMAC-SHA2-256 ☐ Other(s) Configuration(s) - Please indicate which one(s)
7	In the case of implementing the <i>FrodoKEM</i> cryptographic construction, please indicate its parameterization. ☐ Non-Ephemeral ☐ Ephemeral In case both apply, indicate which parameterization applies to each. ☐ AES ☐ SHAKE128 ☐ FrodoKEM-640 ☐ FrodoKEM-976 ☐ FrodoKEM-1344 ☐ Other(s) Configuration(s) - Please indicate which one(s)
8	In the case of implementing the <i>ML-KEM</i> cryptographic construction, please indicate its parameterization. ☐ Supported Configuration(s) - Please indicate which one(s)
9	In the case of implementing any key establishment scheme, please provide the cryptographic mechanisms implemented to authenticate both parties and the data exchanged during the key establishment scheme. Note: Please, provide evidence of the cryptographic mechanisms' implementation (e.g., a pointer to the source code, a piece of source code, etc.). If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
10	Does the TOE implement a self-test KAT for each implemented <i>Key Establishment</i> cryptographic construction (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each key establishment cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).

Q-ID	KeyEstablishment-VQ
11	Does the TOE implement a self-test KAT for each implemented <i>Key Establishment</i> configuration (finite field group, elliptic curve, etc.) (even if they are not in use)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for each configuration (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
12	Does the TOE implement a self-test KAT for each underlying mechanism (key derivation function, XOF primitive, etc.)? Note: In affirmative case, please, indicate the underlying mechanisms that are being tested, describe how they are implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented. If there is more than one implementation for the same construction, please provide evidence of each of them (on different rows).
13	Please provide a brief description of the purpose of each implemented <i>Key Establishment</i> cryptographic construction with its associated parameterization (finite field group, elliptic curve, etc.). If there is more than one implementation for the same construction, please provide the purpose of each of them (on different rows). (e.g., The TOE implements TLS 1.2 whose cipher suites rely on ECDHE with P-256 curve to perform key establishment) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

3.5 RANDOM NUMBER GENERATORS

3.5.1 TRUE RANDOM NUMBER GENERATORS

11. **Note:** In case of multiple TRNG implementations, please complete the following table for each of them, in order to clearly define them. Add as many tables as necessary.

Q-ID	TRNG-VQ
1	Does the TOE implement <i>True Random Number Generators</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	To which AIS-31 functionality class does the <i>TRNG</i> implemented by the TOE belong? Note: The vendor shall provide all the required pieces of evidence in the RNG Vendor Questionnaire.
	☐ PTG.2 ☐ PTG.3 ☐ NTG.1
3	Please provide a brief description of the purpose of the <i>TRNG</i> implemented in the TOE, indicating the SSPs that can be generated using it and the functionality class to which it belongs. (e.g., the TRNG₁-PTG.2 is used to seed and reseed the HMAC-DRBG-DRG.3)
4	In the case of implementing a <i>Physical TRNG (PTRNG)</i>, please provide a schematic of the design of the TOE, showing the TRNG within it.
5	In the case of implementing a <i>Non-Physical TRNG (NPTRNG)</i>, please provide a list enumerating and describing all the source files that conform to it.

3.5.2 DETERMINISTIC RANDOM NUMBER GENERATOR CONSTRUCTION

12. **Note:** In case of multiple DRNG parameterizations and/or implementations, please complete the following table for each of them, in order to clearly define them. Add as many tables as necessary.

Q-ID	DRNG-VQ
1	Does the TOE implement <i>DRNG</i> cryptographic constructions? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>DRNG</i> cryptographic schemes implemented in the TOE?
	☐ HMAC-DRBG ☐ NR (Non-PredictionResistance and Non-Reseed) ☐ NPR (Non-PredictionResistance but with Reseed) ☐ PR (PredictionResistance and Reseed)
	☐ Hash-DRBG ☐ NR (Non-PredictionResistance and Non-Reseed) ☐ NPR (Non-PredictionResistance but with Reseed) ☐ PR (PredictionResistance and Reseed)
	☐ CTR-DRBG (AES) ☐ NR with DF (Non-PredictionResistance and Non-Reseed with Derivation Function) ☐ NR without DF (Non-PredictionResistance and Non-Reseed without Derivation Function) ☐ NPR with DF (Non-PredictionResistance with Reseed and Derivation Function) ☐ NPR without DF (Non-PredictionResistance with Reseed but without Derivation Function) ☐ PR with DF (PredictionResistance and Reseed with Derivation Function) ☐ PR without DF (PredictionResistance and Reseed but without Derivation Function)
	☐ Other(s) - Please indicate which one(s)

Q-ID	DRNG-VQ
3	To which AIS-20 functionality class does the implemented <i>DRNG</i> belong? Note: The vendor shall provide all the required pieces of evidence in the RNG Vendor Questionnaire.
	☐ DRG.3 ☐ DRG.4
4	Which TRNG is used for the seeding and reseeding of the <i>DRNG</i>? Note: Please, provide justification and evidence of the implementation of the seeding and reseeding processes (e.g., a pointer to the source code, a piece of source code, etc.).
5	Does the implemented <i>DRNG</i> provide perfect forward secrecy (PFS)? Note: Justify the answer either in the affirmative or negative case. If the implemented <i>DRNG</i> belongs to AIS-20 functionality class DRG.3 or DRG.4 no additional justification is needed.
5	Does the TOE implement a self-test KAT for the implemented <i>DRNG</i> cryptographic construction (even if they are not in use), that executes all <i>DRNG</i> functions (instantiate, generation and reseed)? Note: In affirmative case, please, describe how it is implemented and provide evidence of the implementation for the <i>DRNG</i> cryptographic construction (e.g., a pointer to the source code, a piece of source code, etc.). It is required to provide the test vectors used in each of the self-test KATs implemented.
6	Does the TOE implement a self-test KAT for each underlying mechanism (hash primitive, AES, HMAC (including correct and incorrect MAC values) construction)? Note: In affirmative case, please, indicate the underlying mechanism(s) that is (are) being tested, describe how it is (they are) implemented, and provide evidence of the implementation. It is required to provide the test vectors used in each of the self-test KATs implemented.
7	Please provide a brief description of the purpose of the implemented <i>DRNG</i> cryptographic construction, indicating the SSPs that can be generated using it and the functionality class to which it belongs. (e.g., the HMAC-DRBG-DRG.3 is used to generate the AES-256 bits key that is used by the TOE to encrypt the information entered through the data input interface) Note: Please, provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).

Q-ID	DRNG-VQ

4 DETERMINISTIC RANDOM NUMBER IMPLEMENTATION

13. This section contains the questions associated with the deterministic random number generators implemented in the TOE.
14. **Note:** In case of multiple DRNG parameterizations and/or implementations, please complete the following tables for each of them, in order to clearly define them. Add as many tables as necessary.

4.1 GENERAL CONSIDERATIONS

Q-ID	General-VQ
1	Is the internal state of the <i>DRNG</i> cryptographic construction [DRNG-State] protected as a CSP in the TOE? Note: In affirmative case, please indicate how it is protected to ensure that it can only be accessed by the DRNG.
2	According to the zeroization and sensitive data requirements, the random data generated by the <i>DRNG</i> cryptographic construction must be properly zeroized once it is no longer needed. Note: Please indicate how the sensitive data specified in “Annex C: List of Temporally Sensitive Data Managed by the TOE” for the implemented DRNG cryptographic construction is being zeroized.

4.2 DRNG INITIALIZATION

Q-ID	Initialization-VQ
1	A <i>DRNG</i> secure initialization depends on the number of bits of the Input (seed), Nonce, and Personalization String parameters. Note: Please indicate the length in bits of each of them during the DRNG initialization.
	The input (seed) length is: XXX bits
	The nonce length is: XXX bits
	The personalization string length is: XXX bits
2	Please provide evidence of the total amount of bits used as <i>Input (seed)</i> and how they are generated. (e.g., a pointer to the source code, a piece of source code, etc.)

Q-ID	Initialization-VQ
3	Please provide evidence of the total amount of bits used as <i>Nonce</i> and how they are generated. (e.g., a pointer to the source code, a piece of source code, etc.)
4	Please provide evidence of the total amount of bits used as <i>Personalization String</i> and how they are generated. (e.g., a pointer to the source code, a piece of source code, etc.)

4.3 DRNG RESEEDING CONFIGURATION

Q-ID	Reseeding-VQ
1	The <i>DRNG</i> reseeding depends on the number of bits used as <i>Input Reseed</i> and as <i>Additional Input</i> in every call to generate a new CSP. Note: Please indicate the length in bits of each of them during the reseeding.
	The input reseed length is: XXX bits
	The additional input length is: XXX bits
2	Please provide evidence of the total amount of bits used for <i>Input Reseed</i> and how they are generated. (e.g., a pointer to the source code, a piece of source code, etc.)
3	Please provide evidence of the total amount of bits used as <i>Additional Input</i> data and how they are generated. (e.g., a pointer to the source code, a piece of source code, etc.)

5 SENSITIVE SECURITY PARAMETERS MANAGEMENT

15. This section contains the questions associated with the sensitive security parameters (SSPs) of the TOE. It aims to provide the tester with all the needed information to demonstrate that the SSP management complies with requirements from Section 2 “CCN Cryptographic Requirements” of the [CCN-MEMeC] methodology.

5.1 SENSITIVE SECURITY PARAMETER GENERATION AND ESTABLISHMENT

Q-ID	Generation-VQ
1	Does the module support <i>SSP Generation</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	If the TOE supports <i>SSP Generation</i>, please indicate the list of Random Number Generators (RNGs), Key Establishment constructions, and Key Derivation Functions (KDFs) implemented in the TOE for their generation.
	☐ RNG – Please indicate which one(s) In the case of asymmetric key generation, complete the associated tables below (RSAKeys-VQ, FFDLOGKeys-VQ, and ECDLOGKeys-VQ).
	☐ Key Establishment – Please indicate which one(s) and the underlying KDF(s)
	☐ Key Derivation Function – Please indicate which one(s)
	☐ Other(s) - Please indicate which one(s)
3	For each SSP generated by the TOE using each of the specified approved cryptographic mechanisms, please complete the table in “Annex B: List of SSPs Managed by the TOE”.
4	For each SSP included in the tables in “Annex B: List of SSPs Managed by the TOE”, please provide evidence of the approved cryptographic mechanisms used for their generation. (e.g., a pointer to the source code, a piece of source code, etc.)

16. For each implementation for which RSA keys are generated, please complete the following table for each of them, in order to clearly define them:

Q-ID	RSASKeys-VQ
1	Does the TOE implement RSA key pair generation? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the RSA key generation schemes implemented by the TOE? Note: For each scheme implemented, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ RSA Key-Pair Generation – Algorithm 6, according to [CCN-STIC-221] guide ☐ Other(s) - Please indicate which one(s)
3	What methods for prime random generation are employed by the implemented RSA key generation schemes? Note: For each method employed, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ “Method 1” – Algorithm 3, according to [CCN-STIC-221] guide ☐ “Method 2” – Algorithm 4, according to [CCN-STIC-221] guide ☐ Other(s) - Please indicate which one(s)
4	What primality tests are implemented in the TOE for each prime random generation method? Note: For each primality test employed, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ Miller-Rabin – Algorithm 5, according to [CCN-STIC-221] guide ☐ Other(s) - Please indicate which one(s)
5	What are the random uniform schemes used by the TOE for the generation of the testing candidates? Note: For each scheme employed, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ Testing Technique – Algorithm 1, according to [CCN-STIC-221] guide

	☐ Extra Random Technique – Algorithm 2, according to [CCN-STIC-221] guide
	☐ Other(s) - Please indicate which one(s)
6	What are the <i>random uniform</i> schemes used by the TOE for the generation of the basis <i>a</i> needed by the <i>Miller-Rabin</i> algorithm? Note: For each scheme employed, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ Testing Technique – Algorithm 1, according to [CCN-STIC-221] guide
	☐ Extra Random Technique – Algorithm 2, according to [CCN-STIC-221] guide
	☐ Other(s) - Please indicate which one(s)
7	What is the number of iterations used in the <i>Miller-Rabin</i> algorithm? Note: Please, justify that this number is sufficient according to the [CCN-MEMeC] methodology and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).

17. For each implementation for which FFDLOG keys are generated, please complete the following table for each of them, in order to clearly define them:

Q-ID	FFDLOGKeys-VQ
1	Does the TOE implement <i>FFDLOG</i> key pair generation? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>random uniform</i> schemes used by the TOE for the generation of the private keys? Note: For each scheme employed, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.).

Q-ID	FFDLOGKeys-VQ
	☐ Testing Technique – Algorithm 1, according to [CCN-STIC-221] guide ☐ Extra Random Technique – Algorithm 2, according to [CCN-STIC-221] guide ☐ Other(s) - Please indicate which one(s)
3	Do the implemented <i>random uniform</i> schemes apply to the output of an agreed deterministic random number generator? ☐ Yes ☐ No

18. For each implementation for which ECDLOG keys are generated, please complete the following table for each of them, in order to clearly define them:

Q-ID	ECDLOGKeys-VQ
1	Does the TOE implement <i>ECDLOG</i> key pair generation? Note: If the response is “No”, the following questions in this table shall not be answered. ☐ Yes ☐ No
2	What are the <i>random uniform</i> schemes used by the TOE for the generation of the private keys? Note: For each scheme employed, please describe how it is implemented and provide evidence of the implementation (e.g., a pointer to the source code, a piece of source code, etc.). ☐ Testing Technique – Algorithm 1, according to [CCN-STIC-221] guide ☐ Extra Random Technique – Algorithm 2, according to [CCN-STIC-221] guide ☐ Other(s) - Please indicate which one(s)
3	Do the implemented <i>random uniform</i> schemes apply to the output of an agreed deterministic random number generator? ☐ Yes ☐ No

5.2 SENSITIVE SECURITY PARAMETER ENTRY/OUTPUT

Q-ID	Transport-VQ
1	Does the module support <i>SSP Entry/Output</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	If the TOE supports <i>PSP Entry/Output</i>, please indicate the list of cryptographic mechanisms (CBC-MAC, GMAC, HMAC, Digital Signature, etc.) implemented to protect their authenticity and integrity.
3	If the TOE supports <i>CSP Entry/Output</i>, please indicate the list of cryptographic mechanisms (SIV, AES-KeyWrap, and/or Authenticated Encryption constructions) implemented to protect their authenticity, integrity, and confidentiality.
4	For each <i>SSP entered/output</i> to/from the TOE using each of the specified approved cryptographic mechanisms, please complete the table in “Annex B: List of SSPs Managed by the TOE”.
5	For each SSP included in the table in “Annex B: List of SSPs Managed by the TOE”, please provide evidence of the approved cryptographic mechanisms used for their entry/output. (e.g., a pointer to the source code, a piece of source code, etc.)

5.3 SENSITIVE SECURITY PARAMETER STORAGE

Q-ID	Storage-VQ
1	Does the module support <i>SSP Storage</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	If the TOE supports <i>PSP storage</i>, please indicate the list of cryptographic mechanisms (CBC-MAC, GMAC, HMAC, and Digital Signature) implemented to protect their authenticity and integrity.
3	If the TOE supports <i>CSP Storage</i>, please indicate the list of cryptographic mechanisms (SIV, AES-KeyWrap, and/or Authenticated Encryption constructions) to protect their authenticity, integrity, and confidentiality.
4	For each <i>SSP stored</i> in the TOE using each of the specified approved cryptographic mechanisms, please complete the table in “Annex B: List of SSPs Managed by the TOE”.
5	For each <i>SSP</i> included in the table in “Annex B: List of SSPs Managed by the TOE”, please provide evidence of the approved cryptographic mechanisms used for their storage. (e.g., a pointer to the source code, a piece of source code, etc.)

5.4 SENSITIVE SECURITY PARAMETER ZEROIZATION

Q-ID	Zeroization.1-VQ
1	Does the module support <i>CSP Zeroization</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	If the TOE supports <i>CSP Zeroization</i>, please indicate the list of zeroization methods (overwriting with zeros, ones, or random data) implemented in the TOE. Note: Please, provide evidence of their implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ Overwriting with Zeros ☐ Please provide evidence
	☐ Overwriting with Ones ☐ Please provide evidence
	☐ Overwriting with Random Data ☐ Please provide evidence
3	Does the TOE automatically zeroize the <i>CSPs</i> or require operator intervention? Note: Please provide evidence of the process (e.g., a pointer to the source code, a piece of source code, etc.).
4	For each zeroized <i>CSP</i> using one of the specified approved cryptographic mechanisms, please complete the table in “Annex B: List of SSPs Managed by the TOE” and provide evidence of their zeroization. (e.g., a pointer to the source code, a piece of source code, etc.)
5	Does the TOE support zeroization based on overwriting an old <i>CSP</i> with a new one? Note: In affirmative case, please provide evidence of the process (e.g., a pointer to the source code, a piece of source code, etc.).
6	Is there any way to recover any zeroized <i>CSP</i>? Note: In affirmative case, please provide evidence of the process (e.g., a pointer to the source code, a piece of source code, etc.).

Q-ID	Zeroization.2-VQ
1	Does the TOE zeroize all <i>Temporally Sensitive Data</i> once they are no longer needed? Note: In the affirmative case, please indicate which sensitive data zeroized by the TOE using the list provided in “Annex C: List of Temporally Sensitive Data Managed by the TOE” and provide evidence of zeroization for each of the cryptographic mechanisms (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ Yes ☐ No
2	If the TOE supports <i>Temporally Sensitive Data Zeroization</i>, please indicate the list of zeroization methods (overwriting with zeros, ones, or random data) implemented in the TOE. Note: Please, provide evidence of their implementation (e.g., a pointer to the source code, a piece of source code, etc.).
	☐ Overwriting with Zeros ☐ Please provide evidence
	☐ Overwriting with Ones ☐ Please provide evidence
	☐ Overwriting with Random Data ☐ Please provide evidence
	☐ Other(s) - Please indicate which one(s) and provide evidence
3	Is there any way to recover any zeroized <i>Temporally Sensitive Data</i>? Note: In affirmative case, please provide evidence of the process (e.g., a pointer to the source code, a piece of source code, etc.).
4	Does the TOE support zeroization based on overwriting an old <i>Temporally Sensitive Data</i> parameter with a new one? Note: In affirmative case, please provide evidence of the zeroization process (e.g., a pointer to the source code, a piece of source code, etc.).

6 CRYPTOGRAPHIC MECHANISMS SELF-TEST

19. This section contains the questions associated with the KATs implemented by the TOE during self-tests to demonstrate the correct operativity of all the cryptographic mechanisms supported by the TOE. It aims to provide the tester with all the needed information to demonstrate that the implementation of the self-tests complies with requirements from Section 2 “CCN Cryptographic Requirements” of the [CCN-MEMeC] methodology.

Q-ID	Implementation-VQ
1	<i>Cryptographic Mechanisms Self-Test</i> Please provide a list of the <i>self-test KATs</i> performed by the TOE regarding the cryptographic mechanisms implemented.
2	<i>Cryptographic Mechanisms Self-Test</i> Please describe under what circumstances or conditions the <i>self-test KATs</i> associated with the cryptographic mechanisms implemented in the TOE are performed.

7 MITIGATION OF OTHER ATTACKS

20. This section contains the questions associated with the countermeasures implemented by the TOE to mitigate side-channel attacks and differential fault analysis. It aims to provide the tester with all the needed information to demonstrate that the implementation of the countermeasures complies with requirements from Section 2 “CCN Cryptographic Requirements” of the [CCN-MEMeC] methodology.

Q-ID	Mitigation-VQ
1	Does the TOE implement <i>Countermeasures</i> to mitigate <i>Side-Channel Attacks</i>? Note: In the affirmative case, please provide information and evidence about the countermeasures implemented to mitigate side-channel attacks.
2	Does the TOE implement <i>Countermeasures</i> to mitigate <i>Differential Fault Analysis (DFA)</i>? Note: In the affirmative case, please provide information and evidence about the countermeasures implemented to mitigate DFA.

8 CRYPTOGRAPHIC PROTOCOLS IMPLEMENTATION

21. This section shall contain information related to the protocols implemented by the TOE which rely on cryptographic mechanisms. The proprietary and/or standard (e.g., TLS, SSH, etc.) protocols and their configuration shall be described justifying the ensured properties.
22. **Note:** This is the information that shall be included in each column:
- **Protocol:** The name of the implemented protocol.
 - **Configuration:** The cipher suites (for TLS) or cryptographic mechanisms configuration employed to implement the protocol.
 - **Properties:** The ensured properties with the protocol configuration. Some examples could be:
 - *Integrity protection of the exchanged messages*
 - *Anti-replay protections*
 - *Perfect forward secrecy*
 - *Post compromise secrecy*
 - *Quantum resistance*
 - **Justification:** Justification for each ensured property by the protocol configuration.
23. Please complete the following table with the protocols implemented by the TOE:

Protocol	Configuration	Properties	Justification

Table 1. Cryptographic Protocols Implementation and Configuration

8.1 TRANSPORT LAYER SECURITY PROTOCOL (TLS)

24. This subsection contains the questions associated with the TLS cryptographic protocol configurations implemented in the TOE. It aims to provide the tester with all the needed information to demonstrate that the implementation of the TLS cryptographic protocol complies with the requirements from Section 3 “CCN Agreed Cryptographic Mechanisms” of the [CCN-MEMeC] methodology.

8.1.1 TLS V1.3

Q-ID	TLS-VQ
1	Does the TOE implement <i>TLS v1.3</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the <i>TLS v1.3 Cipher Suites</i> implemented in the TOE?
	☐ TLS_AES_256_GCM_SHA384
	☐ TLS_AES_128_GCM_SHA256
	☐ TLS_AES_128_CCM_SHA256
	☐ TLS_CHACHA20_POLY1305_SHA256
	☐ Other(s) - Please indicate which one(s)
3	Please provide a brief description of the purpose of each <i>TLSv1.3</i> implementation with its associated cipher suites (on different rows). (e.g., The TOE implements <i>TLSv1.3</i> with the cipher suite <i>TLS_AES_256_GCM_SHA384</i> to establish secure channels with the server) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

25. **Note:** Please complete the following table for each *TLS v1.3* cipher suite implemented, where [ID] means the cipher suite name. Add as many tables as necessary.

Q-ID	TLSv1.3 CIPHER SUITE – [ID]
	Please indicate the implemented <i>TLS v1.3 PSK</i> modes (if supported).

Q-ID	TLSv1.3 CIPHER SUITE – [ID]
1	☐ psk_ke ☐ psk_dhe_ke
	☐ Other(s) - Please indicate which one(s)
2	In the case of implementing TLS v1.3 PSK modes, please provide evidence that the TOE does not allow the sending of zero Round-Trip Time data or 0-RTT data. (e.g., a pointer to the source code, a piece of source code, etc.)
3	Please indicate the implemented DH groups for TLS v1.3 Key Establishment.
	☐ ffdhe2048 ☐ ffdhe3072
	☐ ffdhe4096
	☐ Other(s) - Please indicate which one(s)
4	Please indicate the implemented ECDH groups for TLS v1.3 Key Establishment.
	☐ BrainpoolP256r1tls13 ☐ P-256 (secp256r1)
	☐ BrainpoolP384r1tls13 ☐ P-384 (secp384r1)
	☐ BrainpoolP512r1tls13 ☐ P-521 (secp521r1)
	☐ Other(s) - Please indicate which one(s)
5	Please indicate the implemented xECDH groups for TLS v1.3 Key Establishment.
	☐ x25519 ☐ x448
6	Please indicate the implemented TLS v1.3 Digital Signature Schemes to perform client-server digital signature.
	☐ rsa_pss_rsae_sha256 ☐ rsa_pss_rsae_sha384
	☐ rsa_pss_rsae_sha512 ☐ rsa_pss_pss_sha256
	☐ rsa_pss_pss_sha384 ☐ rsa_pss_pss_sha512
	☐ ecdsa_brainpoolP256r1tls13_sha256 ☐ ecdsa_secp256r1_sha256
	☐ ecdsa_brainpoolP384r1tls13_sha384 ☐ ecdsa_secp384r1_sha384
	☐ ecdsa_brainpoolP512r1tls13_sha512 ☐ ecdsa_secp521r1_sha512
☐ Ed25519 ☐ Ed448	

Q-ID	TLSv1.3 CIPHER SUITE – [ID]
	☐ Other(s) - Please indicate which one(s)
7	Please indicate the implemented TLS v1.3 Digital Signature Schemes on certificates.
	☐ rsa_pkcs1_sha256 ☐ rsa_pkcs1_sha384
	☐ rsa_pkcs1_sha512 ☐ rsa_pss_rsae_sha256
	☐ rsa_pss_rsae_sha384 ☐ rsa_pss_rsae_sha512
	☐ rsa_pss_pss_sha256 ☐ rsa_pss_pss_sha384
	☐ rsa_pss_pss_sha512
	☐ ecdsa_brainpoolP256r1tls13_sha256 ☐ ecdsa_secp256r1_sha256
	☐ ecdsa_brainpoolP384r1tls13_sha384 ☐ ecdsa_secp384r1_sha384
	☐ ecdsa_brainpoolP512r1tls13_sha512 ☐ ecdsa_secp521r1_sha512
	☐ Ed25519 ☐ Ed448
☐ Other(s) - Please indicate which one(s)	

8.1.2 TLS V1.2

Q-ID	TLS-VQ
1	Does the TOE implement TLS v1.2? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	What are the TLS v1.2 Cipher Suites implemented in the TOE?
	☐ TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
	☐ TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
	☐ TLS_ECDHE_ECDSA_WITH_AES_256_CCM
	☐ TLS_ECDHE_ECDSA_WITH_AES_128_CCM
	☐ TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256

Q-ID	TLS-VQ
	☐ TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 ☐ TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 ☐ TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 ☐ TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 ☐ TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 ☐ TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 ☐ TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 ☐ TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 ☐ TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 ☐ TLS_DHE_RSA_WITH_AES_256_CCM ☐ TLS_DHE_RSA_WITH_AES_128_CCM ☐ TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 ☐ TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 ☐ TLS_RSA_WITH_AES_256_GCM_SHA384 ☐ TLS_RSA_WITH_AES_128_GCM_SHA256 ☐ TLS_RSA_WITH_AES_256_CCM ☐ TLS_RSA_WITH_AES_128_CCM ☐ TLS_RSA_WITH_AES_256_CBC_SHA256 ☐ TLS_RSA_WITH_AES_128_CBC_SHA256 ☐ TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 ☐ TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 ☐ TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 ☐ TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256 ☐ TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384

Q-ID	TLS-VQ
	☐ TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256 ☐ TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384 ☐ TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256 ☐ TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256 ☐ TLS_DHE_PSK_WITH_AES_128_CBC_SHA256 ☐ TLS_DHE_PSK_WITH_AES_256_CBC_SHA384 ☐ TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 ☐ TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 ☐ TLS_DHE_PSK_WITH_AES_128_CCM ☐ TLS_DHE_PSK_WITH_AES_256_CCM ☐ Other(s) - Please indicate which one(s)
3	In the case of implementing a TLS cipher suite whose encryption mechanisms are based on CBC, please provide evidence that they are used in conjunction with the “encrypt_then_mac” extension. (e.g., a pointer to the source code, a piece of source code, etc.)
4	Please provide a brief description of the purpose of each TLSv1.2 implementation with its associated cipher suites (on different rows). (e.g., The TOE implements TLSv1.2 with the cipher suite TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 to establish secure channels with the server) Note: Please, provide evidence of each implementation (e.g., a pointer to the source code, a piece of source code, etc.).

26. **Note:** Please complete the following table for each TLS v1.2 cipher suite implemented, where [ID] means the cipher suite name. Add as many tables as necessary.

Q-ID	TLSv1.2 CIPHER SUITE – [ID]
	Please indicate the implemented DH groups for <i>TLS v1.2 Key Establishment</i> .

Q-ID	TLSv1.2 CIPHER SUITE – [ID]			
1	☐ ffdhe2048 ☐ ffdhe3072			
	☐ ffdhe4096			
	☐ Other(s) - Please indicate which one(s)			
2	Please indicate the implemented ECDH groups for <i>TLS v1.2 Key Establishment</i> .			
	☐ BrainpoolP256r1tls13 ☐ P-256 (secp256r1)			
	☐ BrainpoolP384r1tls13 ☐ P-384 (secp384r1)			
	☐ BrainpoolP512r1tls13 ☐ P-521 (secp521r1)			
	☐ Other(s) - Please indicate which one(s)			
3	Please indicate the implemented xECDH groups for <i>TLS v1.2 Key Establishment</i> .			
	☐ x25519 ☐ x448			
4	Please indicate the implemented <i>TLS v1.2 Digital Signature Schemes and Hash Functions</i> .			
		RSA	DSA	ECDSA
	SHA2-256	☐	☐	☐
	SHA2-384	☐	☐	☐
	SHA2-512	☐	☐	☐
	☐ Ed25519 ☐ Ed448			
☐ Other(s) - Please indicate which one(s)				

8.2 SECURE SHELL (SSH)

27. This subsection contains the questions associated with the SSH cryptographic protocol configurations implemented in the TOE. It aims to provide the tester with all the needed information to demonstrate that the implementation of the SSH cryptographic protocol complies with the requirements from Section 3 “CCN Agreed Cryptographic Mechanisms” of the [CCN-MEMeC] methodology.
28. **Note:** In case of multiple SSH implementations, please complete the following table for each of them, in order to clearly define them. Add as many tables as necessary.

Q-ID	SSH-VQ
1	Does the TOE implement SSH-2 protocol? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	Please indicate the implemented DH groups for SSH-2 Key Establishment.
	☐ DH-exchange-SHA256 ☐ DH-group15-SHA512
	☐ DH-group16-SHA512 ☐ DH- group17-SHA512
	☐ DH- group18-SHA512
	☐ Other(s) - Please indicate which one(s)
3	In the case of implementing the <i>Diffie-Hellman-group-exchange-SHA256</i> for SSH-2 Key Establishment, please provide the following information:
	The prime p and its length in bits: XXX
	The generator g : XXX
	The prime p and its length in bits: XXX
4	Please indicate the implemented ECDH groups for SSH-2 Key Establishment.
	☐ secp256r1_sha512 ☐ secp384r1_sha512
	☐ secp521r1_sha512
	☐ Other(s) - Please indicate which one(s)
5	Please indicate the implemented SSH-2 Encryption Mechanisms.
	☐ AEAD_AES_128_GCM ☐ AEAD_AES_256_GCM

Q-ID	SSH-VQ
	☐ AES128-CTR ☐ AES192-CTR ☐ AES256-CTR ☐ Other(s) - Please indicate which one(s)
6	Please indicate the implemented SSH-2 Integrity and Authenticity Mechanisms. ☐ HMAC-SHA1 ☐ HMAC-SHA2-256 ☐ HMAC-SHA2-512 ☐ AEAD_AES_128_GCM ☐ AEAD_AES_256_GCM ☐ Other(s) - Please indicate which one(s)
7	Please indicate the implemented SSH-2 Server-Client Authentication Mechanisms. ☐ ECDSA-SHA2-secp256r1 ☐ ECDSA-SHA2-secp384r1 ☐ ECDSA-SHA2-secp521r1 ☐ x509v3-ECDSA-SHA2-secp256r1 ☐ x509v3-ECDSA-SHA2-secp384r1 ☐ x509v3-ECDSA-SHA2-secp521r1 ☐ Other(s) - Please indicate which one(s)
8	Please indicate whether the SSH-2 session keys are renewed after an hour or after the transmission of one gigabyte of data, whichever occurs first. Note: Justify the answer either in the affirmative or negative case.
9	Please provide a brief description of the purpose of the SSH-2 implementation with its associated configurations. (e.g., The TOE implements SSH-2 to allow remote access to users) Note: Please, provide evidence of the SSH implementation (e.g., a pointer to the source code, a piece of source code, etc.).

8.3 INTERNET PROTOCOL SECURITY (IPSEC) WITH IKEV2

29. This subsection contains the questions associated with the IPsec cryptographic protocol configurations implemented in the TOE. It aims to provide the tester with all the needed information to demonstrate that the implementation of the IPsec cryptographic protocol complies with the requirements from Section 3 “CCN Agreed Cryptographic Mechanisms” of the [CCN-MEMeC] methodology.
30. **Note:** In case of multiple IPsec implementations, please complete the following table for each of them, in order to clearly define them. Add as many tables as necessary.

Q-ID	IPsec-VQ
1	Does the TOE implement <i>IPsec</i> protocol with <i>IKEv2</i>? Note: If the response is “No”, the following questions in this table shall not be answered.
	☐ Yes ☐ No
2	Please indicate the implemented <i>Protocol Versions</i>.
	☐ IPsec ☐ IKEv2 ☐ ESP
3	Please indicate the implemented <i>IPsec</i> mode.
	☐ Tunnel Mode ☐ Transport Mode
4	Please indicate the implemented DH groups implemented for the <i>IKEv2</i> Key Establishment.
	☐ 3072-bit MODP ☐ 4096-bit MODP ☐ 6144-bit MODP ☐ 8192-bit MODP ☐ Other(s) - Please indicate which one(s)
5	Please indicate the implemented ECDH groups implemented for the <i>IKEv2</i> Key Establishment.
	☐ 256-bit random ECP ☐ 384-bit random ECP ☐ 521-bit random ECP ☐ brainpoolP256r1 ☐ brainpoolP384r1 ☐ brainpoolP512r1 ☐ Other(s) - Please indicate which one(s)

Q-ID	IPsec-VQ
6	Please indicate the implemented xECDH groups implemented for the IKEv2 Key Establishment.
	☐ x25519 ☐ x448
	☐ Other(s) - Please indicate which one(s)
7	Please indicate the implemented IKEv2 and ESP Encryption Mechanisms.
8	Please indicate the implemented IKEv2 and ESP Origin Integrity and Authenticity Mechanisms.
	☐ HMAC-SHA2-256_128 ☐ HMAC-SHA2-384_192
	☐ HMAC-SHA2-512_256 ☐ AES-CMAC-96
	☐ AES-GMAC
	☐ Other(s) - Please indicate which one(s)
9	Please indicate the implemented IKEv2 Pseudo-Random Functions.
	☐ HMAC-SHA2-256 ☐ HMAC-SHA2-384
	☐ HMAC-SHA2-512 ☐ AES128-CMAC
	☐ Other(s) - Please indicate which one(s)
10	Please indicate the implemented IKEv2 Authentication Mechanisms.
	☐ RSA (RSASSA-PSS) with SHA-256 ☐ RSA (RSASSA-PSS) with SHA-384
	☐ RSA (RSASSA-PSS) with SHA-512 ☐ ECDSA-SHA-256 and P-256
	☐ ECDSA-SHA-384 and P-384 ☐ ECDSA-SHA-512 and P-521
	☐ ECDSA-256-BrainpoolP256r1 ☐ ECDSA-384-BrainpoolP384r1
	☐ ECDSA-512-BrainpoolP512r1 ☐ ECGDSA-256-BrainpoolP256r1
	☐ ECGDSA-384-BrainpoolP384r1 ☐ ECGDSA-512-BrainpoolP512r1
	☐ Other(s) - Please indicate which one(s)

Q-ID	IPsec-VQ
11	Please provide a brief description of the purpose of each <i>IPsec</i> implementation with its associated configurations. (e.g., The TOE implements IPsec to establish secure channels over a VPN) Note: Please, provide evidence of the IPsec implementation (e.g., a pointer to the source code, a piece of source code, etc.).

9 ANNEX A: SENSITIVE SECURITY PARAMETERS DEFINITION

31. This section contains the sensitive security parameters (SSPs) of the TOE that shall be cryptographically protected according to Section 5 “[Sensitive Security Parameters Management](#)”. SSPs can be divided into two main categories:
- a) Critical Security Parameters (CSPs): shall be protected in confidentiality, integrity, and authenticity.
 - b) Public Security Parameters (PSPs): shall be protected in integrity and authenticity.
32. The following table provides the list of CSPs associated with each cryptographic mechanism approved by the CCN according to the [CCN-MEMeC] methodology. CSPs are mainly composed of secret and private keys of the cryptographic mechanisms, so although they are part of the sensitive data related to the mechanisms, they are managed differently as they require additional protection to that of the sensitive data obtained as a result of the execution of the cryptographic mechanisms.

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	
	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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Critical Security Parameter (CSP) Description
	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 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	
	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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Critical Security Parameter (CSP) Description
	DSA	The private component of ECDLOG key pairs in digital signature generation operations
	EC-KCDSA	
	EC-DSA	
	EC-GDSA	
	EC-Schnorr	
	EdDSA	The private component of key pairs in digital signature generation operations
	XMSS	
	ML-DSA	
	Falcon	
	SLH-DSA	
Key Establishment Scheme	DH	Shared Secret obtained as a result of the key establishment procedure
	EC-DH	
	DLIES-KEM	
	ECIES-KEM	The private component of key pairs in secret sharing computation operations
	ML-KEM	
	FrodoKEM	
Deterministic Random Number Generator Construction	HMAC-DRBG	Seed used to initialize or re-seed the DRNG
	Hash-DRBG	The internal state of DRNG generated in each generation operation
	CTR-DRBG (AES)	

Table 2. CSPs definition for each Cryptographic Mechanism

33. The following table provides the list of PSPs associated with each cryptographic mechanism approved by the CCN according to the [CCN-MEMeC] methodology.

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	
	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 3. PSPs definition for each Cryptographic Mechanism

10 ANNEX B: LIST OF SSPS MANAGED BY THE TOE

34. This section contains the list of SSPs managed by the TOE, specifying the cryptographic mechanisms used for their generation, their protection during entry into/output from the TOE, their storage, zeroization, and purpose.
35. Depending on the evaluation level, the following evidence of SSP management is required:
- For **CL2 evaluations**, it is only required evidence of the zeroization process.
 - For **CL3 evaluations**, evidence of the entire SSP lifecycle (generation, entry/output, storage, application, and zeroization processes) is required.
36. **Note:** This is the information that shall be included in each column:
- **SSP:** The name of the SSP.
 - **Strength:** The length in bits of the generated SSP.
 - **Generation:** The agreed cryptographic mechanisms used in the case of generating the SSPs into the TOE.
Please indicate the mechanisms involved in the complete generation process, from the TRNG to the CSP generation method. For example, for asymmetric keys based on RSA: TRNG₁-PTG.2 – HMAC-DRBG-DRG.3 – Algorithm 6 of RSA Key-Pair Generation.
 - **Entry/Output:** The agreed cryptographic mechanisms used to protect the SSPs in the case they are entered into/output from the TOE.
 - **Storage:** The agreed cryptographic mechanisms used to protect the SSPs stored within the TOE. In addition, it is necessary to specify the storage location of the SSP (for example, in RAM memory).
 - **Application or Cryptographic Operation:** Definition of the purpose of each SSP and the cryptographic mechanisms related to it.
 - **Zeroization:** Agreed zeroization technique employed to zeroize the specific SSP.
37. The following table shows an example:

SSP	Strength (in bits)	Generation	Entry/Output	Storage	Application or Cryptographic Operation	Zeroization
AES_EDK	256	Method Generated using a TRNG + DRNG: TRNG1-PTG.2 – HMAC-DRBG-DRG.3	Method N/A	Method Secure storage in persistent memory using AES-256 KeyWrap	Application or Crypto Operation Authenticated encryption/decryption of user data with: AES-256-GCM	Method Overwritten using zeros
		Evidence for CL3 Include a piece of source code and justification	Evidence for CL3 N/A	Evidence for CL3 Include a piece of source code and justification	Evidence for CL3 Include a piece of source code and justification	Evidence for CL2 and CL3 Include a piece of source code and justification

Table 4. Example of the SSP Management table

38. Please complete the following table with the SSPs managed by the TOE:

SSP	Strength (in bits)	Generation	Entry/Output	Storage	Application or Cryptographic Operation	Zeroization
		Method	Method	Method	Application or Crypto Operation	Method
		Evidence for CL3	Evidence for CL3	Evidence for CL3	Evidence for CL3	Evidence for CL2 and CL3
		Method	Method	Method	Application or Crypto Operation	Method
		Evidence for CL3	Evidence for CL3	Evidence for CL3	Evidence for CL3	Evidence for CL2 and CL3

Table 5. SSP Management

11 ANNEX C: LIST OF TEMPORALLY SENSITIVE DATA MANAGED BY THE TOE

39. This section contains the temporally sensitive data different from the CSPs and SSPs detailed in the previous section, whose disclosure could compromise the operation of a cryptographic mechanism. Therefore, they shall be protected by the TOE in the sense of confidentiality and/or integrity to prevent disclosure to an attacker using advanced techniques such as Side-Channel attacks.
40. Please complete the following table with the temporally sensitive data managed by the TOE and the evidence of zeroization:

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
Block Cipher Primitive	AES	Sensitive data that shall be zeroized because it would permit the derivation of the encryption key: ☐ Each round key in the AES algorithm execution ☐ Erroneous ciphertexts ☐ Plaintext on the encryption side ☐ Temporally variables such as the state and counter of each round ☐ Initialization vectors on the encryption side ☐ Counters on the encryption side	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
Stream Cipher Primitive	ChaCha20	Sensitive data that shall be zeroized: ☐ 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	Sensitive data that shall be zeroized: ☐ 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	Sensitive data that shall be zeroized: ☐ Each of the parts (specific to each user) that are needed to generate the shared secret	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
XOF	SHAKE cSHAKE	Sensitive data that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	AES-CFB	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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 ☐ 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 that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	AES-OFB	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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 that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
		☐ 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 ☐ 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 that shall be zeroized: ☐ 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 that shall be zeroized: ☐ The input message and the generated MAC ☐ Results of the XOR operations internally performed	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	AES-GMAC	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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 that shall be zeroized: ☐ The input message and the generated MAC ☐ Internal states during the execution	
	Poly1305	Sensitive data that shall be zeroized: ☐ The input message and the generated MAC	
	Encrypt-then-MAC	Sensitive data related to the underlying AES block cipher primitive and the underlying	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
Authenticated Encryption Construction		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	
	AES-GCM	Sensitive data related to the AES block cipher primitive shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	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 cipher primitive and the Poly1305 cryptographic 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	Evidence of Zeroization and Explanation
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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	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	
	PBKDF2	Sensitive data related to the HMAC construction shall be zeroized No additional sensitive data related to this scheme	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	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 Construction	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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	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	
Integer Factorization Primitive	RSA	Sensitive data that shall be zeroized: ☐ 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	Evidence of Zeroization and Explanation
Multiplicative Discrete Logarithm Problem	FFDLOG	Sensitive data that shall be zeroized: ☐ 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 as sensitive data: ☐ The base(s) ☐ The exponent(s) ☐ The result	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
Additive Discrete Logarithm Problem	ECDLOG	Sensitive data that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
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 that shall be zeroized: ☐ The plaintext to be encrypted in the encryption operation ☐ The cyphertext (encoded plaintext) in the case of decryption operations ☐ Output of the hash function employed ☐ Intermediate values for OAEP encoding and decoding operations	
	RSA-PKCS#1v1.5	Sensitive data related to the RSA asymmetric primitive shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ The plaintext to be encrypted in the encryption operation ☐ The cyphertext (encoded plaintext) in the case of decryption operations	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
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 that shall be zeroized: ☐ 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 that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
		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	
	EC-DSA EC-KCDSA EC-GDSA EC-Schnorr	Sensitive data related to the ECDLOG asymmetric primitive and SHA hash primitive shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
		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 Additional sensitive data related to this scheme that shall be zeroized: ☐ 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	Evidence of Zeroization and Explanation
	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 ☐ The current state information and the OTS private keys ☐ All the intermediate result values in the <i>wots+</i> algorithm (such as the message in base <i>w</i> or the checksum) ☐ All sensitive parameters of the pseudo-random function	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	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: ☐ The message to be signed is generally considered public data, however, in some situations it needs to be kept secret. In that	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
		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 that shall be zeroized: ☐ The random secret values used as the exponent in DH to produce the shared secret and all the data derived from the base secret	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	EC-DH	Sensitive data related to the ECDLOG asymmetric primitive and the KDF construction shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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 KDF cryptographic construction shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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 sensitive data. Moreover, the data derived during the operation is considered sensitive data	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
	ECIES-KEM	Sensitive data related to the ECDLOG asymmetric primitive and KDF cryptographic construction shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
Deterministic Random Number Generator Construction	HMAC-DRBG	Sensitive data related to the HMAC construction shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ 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 primitive shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ The random bit output by the DRNG mechanism is often considered sensitive data because they are usually used to generate secret keys	
	CTR-DRBG (AES)	Sensitive data related to the AES block cipher primitive and AES-CTR construction shall be zeroized Additional sensitive data related to this scheme that shall be zeroized: ☐ The random bit output by the DRNG mechanism is often considered sensitive data	

Cryptographic Mechanism	Cryptographic Mode/Scheme	Temporally Sensitive Data	Evidence of Zeroization and Explanation
		because they are usually used to generate secret keys	

Table 6. Temporally Sensitive Data Parameters for each Cryptographic Mechanism

12 REFERENCES

- [CCN-MEMeC]** CCN Cryptographic Mechanisms Evaluation Methodology v1.3.3. November 2024
- [CCN-STIC-221]** CCN. Guía de Mecanismos Criptográficos autorizados por el CCN v2.1. Octubre 2024.
- [SP800-133]** NIST. Recommendation for Cryptographic Key Generation. National Institute of Standard and Technology, Special Publication SP800-133, Revision 2, 2020.
- [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.

13 ACRONYMS

AE	Authenticated Encryption
CCN	Centro Criptológico Nacional
CL1	Certification Level 1
CL2	Certification Level 2
CL3	Certification Level 3
CSP	Critical Security Parameter
DFA	Differential Fault Analysis
DRBG	Deterministic Random Bit Generator
ECDLOG	Discrete Logarithm in Elliptic Curves
FFDLOG	Discrete Logarithm in Finite Fields
IPsec	Internet Protocol security
KAT	Known Answer Test
KDF	Key Derivation Function
MAC	Message Authentication Code
PSP	Public Security Parameter
RNG	Random Number Generator
SSH	Secure Shell
SSP	Sensitive Security Parameter
TLS	Transport Layer Security Protocol
TOE	Target Of Evaluation
TRNG	True Random Number Generator
VQ	Vendor Questionnaire
XOF	Extendable-Output Function

