European Cybersecurity Certification Group
Sub-group on Cryptography

Agreed Cryptographic Mechanisms

Version draft document
April 2026

This page is intentionally left blank.

Table of contents

1 Introduction

1.1 Objective

This document is primarily addressed to developers and evaluators. Its purpose is to specify which *cryptographic mechanisms* are recognised *agreed*, i.e., ready to be accepted by all national cybersecurity certification authorities (NCCAs).

This document could also help developers, decision makers and users of cryptography to decide which cryptographic mechanisms are state-of-the-art and could cover their need for cryptographic protection, e.g., confidentiality, integrity, data origin authentication and authentication.

This document focuses mainly on security against adversaries interacting with the mechanisms through their standard interface. It contains however advice and caveats related to the implementation of mechanisms, when it is felt that they may be useful to the developers/evaluators and are crucial for security, e.g. typically to warn against implementation errors that are most commonly made.

Agreed cryptographic mechanisms are subdivided into two categories, according to their estimated robustness. This is the confidence placed in their ability to withstand attacks, in the absence of groundbreaking cryptanalytic improvements, e.g., publication of new attacks~~, implementation of a quantum computer~~. For each category of agreed mechanisms, the principles governing the management of the time limits of their validity is summarized below.

- **recommended mechanisms**, that fully reflect the state of the art in cryptography, currently offer a security level of at least 125 bits (see Section 1.3 below for an informal definition of this notion), are supported by strong security arguments and can be said to provide an adequate level of security against presently known or conjectured threats even taking into account the generally expected increases in computing power. The recommended mechanisms are regarded to represent the current state of the art in cryptographic security engineering. Residual threats to their security come from potential groundbreaking developments.
- **admissible mechanisms**, that are deployed on a large scale, ~~currently offer a security level of at least 100 bits~~ and are considered to provide an acceptable short-term security but ~~should be phased out as soon as practical because they~~ do not fully reflect the state of the art and suffer from some security assurance limitations as compared with recommended mechanisms. As a consequence, a validity period is defined for admissible mechanisms.

  Admissible mechanisms that are expected to become vulnerable in the near future should be phased out as soon as practical. The validity period is given by a *deprecation deadline*: the mechanism ceases to be agreed after the given deadline. For instance the notation A[2033] indicates that the acceptance of the mechanism expires on December 31, 2033.

  For admissible mechanisms that are not expected to become vulnerable in the near future, the given validity period is to be interpreted as a minimum validity period~~,~~ for which it is foreseen that the mechanism will remain acceptable. In this case, it may be extended in future versions of this document. The notation A[2033+] indicates that the acceptance of the admissible mechanism goes at least to 2033.

The default validity period for admissible mechanisms is set to A[2033+].

New cryptographic systems should use recommended algorithms and key sizes, and the use of admissible mechanisms in such systems must be justified (for instance by reasons of backward compatibility).

In general, it is impossible to tell how long a cryptographic mechanism will remain secure. It is possible to derive reasonably safe upper bounds to the amount of computing power that will be available to an attacker at some point in the future and it is also possible to conservatively model the impact of the gradual improvement of existing attack ideas on the security of cryptographic mechanisms. However, breakthrough results might still endanger their security.

In case of a significant improvement of relevant cryptanalytic attacks, the ECCG Sub-group on Cryptography will generally downgrade the status of a recommended mechanism to admissible, or speed up the expiration of some agreed admissible mechanisms. In exceptional cases, a decision may need to be made to remove an algorithm from both the recommended and the admissible lists of algorithms in the present document in an accelerated fashion; this is when an improvement in cryptanalytic techniques poses an immediate practical threat to the security of the algorithm in question.

An example for possible improvements in cryptanalytic techniques that will lead to a recommended algorithm being downgraded to admissible status may be the publication of new attacks that show at least a certificational security weakness in the formerly recommended mechanism. For example an attack the complexity of which is clearly lower than the best generic attack.

In order for this document to stay relevant, it will be revised on a two-year basis to take into account the progress of the state of the art.

1.2 Classification of Cryptographic Mechanisms

Algorithms, schemes and protocols. Cryptographic mechanisms are described as *cryptographic algorithms*, that is to say a sequence of instructions to be executed by the system. Among these algorithms, we distinguish *cryptographic schemes*, which are distributed algorithms, involving several parties. We denote by *cryptographic protocols* the cryptographic schemes that are (fully) interactive.

Primitives and constructions. It is customary to distinguish cryptographic mechanisms according to their complexity. Cryptographic mechanisms are usually built upon more elementary mechanisms, denoted as *cryptographic primitives*, in order to achieve higher-level security objectives. When this is the case, we denote them as *cryptographic constructions*. Building cryptographic mechanisms upon (an) established cryptographic primitive(s) enables to derive confidence from the previous analysis of the primitive(s): usually, the security of the cryptographic construction can be expressed in terms of the security of the underlying primitive(s), possibly along with some quantifiable bounded loss of assurance.

A guiding principle in this document is that agreed cryptographic constructions should rely on agreed cryptographic primitives.

Note 1-AgreedPrimitive. In order for a construction to be considered agreed, only agreed primitives shall be used, unless explicitly stated otherwise. Moreover, in order for a construction that is marked "recommended" to result in a recommended mechanism, the underlying primitive(s) shall also be recommended. If it is marked "recommended" but an underlying primitive is marked "admissible", the resulting mechanism is considered "admissible". One has also to keep in mind that general and specific caveat notes related to a type of primitive or (a) specific primitive(s) also apply to constructions involving such (a) primitive(s), even though these notes are not repeated.

Atomic primitives. It must be noted that a given mechanism can be at the same time a construction and a primitive, e.g., ECDSA relies on a cryptographic hash function and the discrete logarithm in a group of points of an elliptic curve, and can be used as a primitive in an authenticated key exchange protocol. A cryptographic mechanism is said to be an *atomic primitive* when it cannot be described as a construction or when such decomposition is not a common practice in the cryptographic industry. Block ciphers like the AES and hash functions like SHA-256 are classified in this category. Note that SHA-256 can be described as a block cipher used in a specific mode of operation. While this description is useful for cryptanalysis purposes, the independent use of said block cipher and mode of operation is not seen in practice: they are firmly tied together. As a consequence, we classify SHA-256 as an atomic cryptographic primitive. The cryptographic robustness of atomic primitives is hard to evaluate, since the difficulty of a mathematical problem has to be assessed. Furthermore, a complete proof of security is generally not achievable: the security status of these mechanisms partly relies on the confidence the evaluator places on academic results. Note that the problems on which most asymmetric schemes rely (RSA problem/integer factorization, finite field discrete logarithm, elliptic curve discrete logarithm) also fall in the category of atomic primitives.

Notes. It must be noted that the disctinctions between cryptographic primitives and constructions, or between cryptographic schemes and cryptographic protocols, that we establish here are somewhat arbitrary, since the boundaries between the concepts may be fuzzy:

- There is a trend in the design of cryptographic mechanisms to favour modular designs, based on an elementary component used in an appropriate mode. For example, SHA-3 is a sponge function built upon a permutation with good cryptographic properties. SHA-3 could thus be considered as much a primitive as a scheme based on this primitive permutation.

- Some types of mechanisms can be obtained either through an atomic primitive, or through a construction, e.g., stream ciphers can be based upon a dedicated stream cipher design or be built upon a block cipher.

- The use of encryption schemes supposes a communication between two parties, one that encrypts the plaintext, another that decrypts the ciphertext. As such, it could be considered as an "elementary protocol", with limited interaction between the parties.

We strive to select an interpretation in line with common practices in the cryptographic community.

Symmetric and Asymmetric. Cryptographic mechanisms generally make use of parameters, denoted *keys*. The security provided by the cryptographic mechanisms relies on the confidentiality and/or the integrity of these keys. Cryptographic schemes define associated operations, that may make use of different associated keys. A cryptographic scheme is said to be *symmetric* when the key used by every party is known or can be derived by the other party (parties). A cryptographic scheme is said to be *asymmetric* when this is not the case. Usually, the keys are identical for all parties in symmetric schemes. Symmetric and asymmetric cryptographic schemes involve quite different designs. By extension, mechanisms are categorized into symmetric or asymmetric according to their design. For example, hash functions are unkeyed cryptographic mechanisms that are considered symmetric cryptographic primitives.

1.3 Security Level

Attack complexities. The security level of a cryptographic mechanism is usually given as the number of operations necessary for an adversary to successfully break the security provided by the mechanism. It is expressed as a base 2 logarithm, e.g., 125 bits of security means that 2^{125} operations are necessary. We can distinguish several complexity metrics, that evaluate the cost of running an attack:

- *Time complexity.* This corresponds to the amount of offline computations required by the attack based on data extracted from the cryptographic mechanism. This can be qualified as the offline complexity. Note that the notion of time complexity is independent of the computational power available to the attacker. If the attack is parallelisable, the attacker can take advantage of this to decrease the wall time of the computation by using more work force, but this does not change the time complexity, i.e., the total amount of computations needed for running the attack. For example, the time complexity of an AES-128 key exhaustive search is (approximately) 2^{128} operations.

- Memory complexity. This traces the amount of storage necessary to perform the attack. For example, Nohl and Krißler's implementation of Barkan, Biham, and Keller's time/memory trade-off on A5/1 requires 2TB of memory to store the rainbow tables.

- Data complexity. This traces the amount of interactions the adversary needs to perform with the target cryptographic mechanism in order to run the attack. It may correspond to the amount of data that the adversary needs to extract from the execution of the target cryptographic mechanisms in order to be able to perform the attack, or the amount of data the adversary has to submit to achieve a given result. This can be qualified as the online complexity. One can further distinguish a passive adversary that only intercepts data, and an active adversary who interacts with the mechanism to collect data corresponding to inputs with special properties. For example, Matsui's DES linear cryptanalysis requires 2^{43} known plaintexts, and Bleichenbacher's PKCS#1v1.5 oracle attack on 1024-bit RSA requires about a million padding oracle queries.

All these complexity measures can also be combined by considering time/memory/data tradeoffs: for example it may be possible to diminish the time complexity of an attack by increasing the memory or data complexity.

Security level. In this document, when the *security level* of a mechanism is mentioned, this is to be implicitly understood as the time complexity of the best known attack (under some assumptions on the adversary's resources). Writing information in the memory and processing data require operations that are assumed here to be taken into account when assessing the time complexity of an attack. With this convention, the security level of a mechanism cannot be lower than the memory complexity, or the data complexity of the best known attack on the mechanism.

When designing or evaluating a cryptographic mechanism, one has to ensure that no attacker can practically break the security of the mechanism. This affects the size of several components of the cryptographic mechanism, e.g., key size and internal state size. Below, we give principles that should guide this part of the evaluation of the cryptographic mechanisms.

1. Recommended mechanisms should provide at least 125 bits of security against offline attacks. ~~100 bits of security are acceptable for legacy mechanisms, but provide a lower security margin.~~
2. In very particular cases, the acceptable offline complexity could be lower. An example would be the case where each long-term key only has a small value, e.g. as in lightweight cryptographic mechanisms implemented in a resource-constrained RFID tag used to prevent theft of a low-value item.
3. Acceptable data complexity figures are lower. Indeed, acquiring data or transmitting data to a process, device, server, that uses some secret key requires to communicate with said device. There are limitations on the communication stemming from the physical limitations of the device (e.g., throughput of a smartcard communication channel), or that can be enforced by the device itself (e.g., number of wrong pins entered before the device is blocked, or time before cryptographic keys expire). Thus it is enough to study the security of the mechanism against attackers who are restricted to a limited amount of data complexity. For example, on a 100 gigabit network, the time necessary to exchange 2^{64} AES blocks is over seven hundred years. Attacks requiring access to more than 2^{64} blocks of data are thus not of practical concern.

Important disclaimer: these levels of acceptable time and data complexities of attacks are given as a guideline. The robustness of a cryptographic mechanism cannot only be reduced to the complexity of the best known attacks, and derives from the confidence gained over a long period of cryptanalysis. An important aspect is how the original security claim of the cryptographic mechanism withstands the scrutiny of cryptanalysts. The discovery of attacks invalidating those claims is a cause of concern, even if the new attacks do not have an immediate practical impact. Attacks only get better.

Key length. An important consequence of the definition of the security level of a mechanism in terms of the minimal amount of computation that has to be performed to break the mechanisms is how this translates in terms of the size of the keys.

In this document, when the notion of key length is mentioned, it has to be understood that it corresponds to the entropy of the key generation mechanism. For a symmetric key stored on k bits, drawn uniformly at random among all the possible keys, the key length is k. Another example is the case of DES keys:

even though the keys of DES are stored on 64 bits, 8 of those bits are redundant, which make the key length of this block cipher only 56 bits. As a consequence the key length of Triple-DES with 3 keys is 168 bits, strictly lower that the 192 bits used to store such keys. For asymmetric keys, the relation between key/parameter length and security level is less straightforward. When a private or ephemeral key is generated, the method of key generation must not be exploitable towards a speedup of the most efficient attacks against the corresponding cryptographic system with uniformly random key generation. In particular, the entropy of these asymmetric keys should be no lower than the acceptable length of a symmetric key.

1.4 Post-Quantum Cryptography

The complexity notions developed in the previous section are relative to a computational model that defines the elementary operations that can be performed. The currently available processor units, built around operation on bits, data units that can assume two values, correspond to the "classical" computation model.

Since the 1980s, a new *quantum* computation model has also been considered. In this model, in addition to classical operations, processing units can work on *qubits* that can assume a superposition of values in an intricated fashion. This computation model has strong implications for cryptography. Indeed, several quantum algorithms developed in the 1990s enable to attack the hardness assumptions that underpin existing cryptographic mechanisms. In particular, the Shor algorithm is a very efficient polynomial-time algorithm that could factor large integers on a large enough quantum computer. While no such cryptographically relevant quantum computer has been implemented yet, the field of quantum computing has gained considerable traction in the past decade. Although we do not yet know if and when the quantum threat will materialize, this situation is sufficient to warrant caution, and to warrant mitigating actions to be taken. In particular, it is important not to postpone mitigating actions in applications where confidentiality is to be protected in the long term since information sent today can be stored by adversaries for decryption in the future, once cryptographically relevant quantum computers become available. This threat is usually referred to as a "harvest now, decrypt later" attack. It is also important to take early mitigating actions for long-lived products aimed at protecting sensitive information.

Cryptographers have been assessing the impact of quantum computing on classical cryptographic mechanisms and developing alternative, *post-quantum*, mechanisms that are conjectured to be resistant against attacks leveraging a large scale quantum computer. Several standardisation processes are ongoing on this topic. Most post-quantum schemes currently proposed for standardisation are relatively new compared to existing schemes, and have received less scrutiny.

The approach recommended in this document is to prioritize mitigating the quantum threat in applications where confidentiality is to be protected in the long term, by rolling out post-quantum secure cryptography in hybrid mode alongside existing classically secure asymmetric cryptography and/or symmetric keying. These hybrid modes shall ensure that all combined pre or post-quantum cryptographic mechanisms need to be broken simultaneously for the hybrid mode to be broken. This approach provides assurance against the quantum threat as well as assurance against security issues that might affect the newer standardized post-quantum mechanisms. In addition to the deployment of hybrid solutions for asymmetric mechanisms, it is also recommended, as a complementary mitigating action, to adapt the parameter lengths of symmetric

mechanisms, e.g., key and hash lengths.

This document introduces several asymmetric post-quantum mechanisms as agreed.

1.5 Organization of the Document

In the body of this document we start by listing cryptographic mechanisms that are considered as agreed cryptographic mechanisms. We describe first symmetric mechanisms, followed by asymmetric mechanisms. For each family, we describe first atomic primitives, followed by cryptographic constructions. Atomic cryptographic primitives are presented by type of mechanisms. Cryptographic constructions are presented according to the type of security objective they achieve.

For each mechanism, the following information is provided:

- a short informal description of the interface, i.e., a description of the inputs and outputs of the mechanism, and functionality of the mechanism, i.e., the generic properties satisfied by the mechanism;
- a table summarising the set of agreed mechanisms, i.e. of all *recommended* or *admissible* mechanisms for the considered type of mechanism. Each mechanism is either marked R (recommended) or A (admissible). If restrictions apply to the parameter values for which the mechanism is considered "R" or "A", these restrictions appear in the "Parameters' size" column of the table;
- caveat notes on how to implement/evaluate the mechanism correctly. General notes apply to all mechanisms of a category, specific notes apply to a more restricted subset of mechanisms.

Secondly, we state some requirements on key management. Indeed, we recall that the security provided by keyed cryptographic mechanisms stems from the appropriate protection of keys, e.g., their confidentiality and integrity. This has strong implications on the key life cycle. This covers aspects like key generation, key usage, or key destruction. We list agreed methods that can be used to generate keys, and express some requirements on other aspects of the key life cycle.

1.6 Related Documents

This document has a specific purpose and is fully self-contained with respect to the specification of which cryptographic mechanisms are considered as agreed by the ECCG Sub-group on Cryptography. It is base on the SOG-IS Agreed Cryptographic Mechanisms document. No automatic compliance with any preexisting list of recommended mechanisms was aimed for. However the content of existing regulatory or advisory documents was taken into account whenever judged relevant. Credit must therefore be given to [ENISAAlgo, TR-02102-1, ANSSI-PG-083] for having partly inspired some of the concepts, definitions, recommendations, or caveats relating to cryptographic mechanisms provided in this document.

2 Symmetric Atomic Primitives

In symmetric cryptography, the security is based on the confidentiality of a key shared by the legitimate users. We start by presenting the agreed symmetric atomic primitives in this section, then describe in the next section the symmetric constructions built upon these primitives.

2.1 Block Ciphers

A block cipher is a family of permutations of $\{0,1\}^n$. A k-bit parameter named the *key* enables to select a given permutation of the family. n and k represent respectively the block size and the key size, expressed in bits.

Agreed Block Ciphers.

Primitive	Parameters' sizes	R/A	Notes
AES [FIPS197, ISO18033-3]	k = 128 bits	A	3-QuantumThreat
	k = 192 bits	R	
	k = 256 bits	R	
Triple-DES [FIPS46-3, ISO18033-3]	k = 168 bits	A[2027]	2-SmallBlocksize 3-QuantumThreat

Specific Notes.

Note 2-SmallBlocksize. Block ciphers are used in modes of operation to provide a wide variety of security functionalities. In the majority of the cases, the modes fulfill their security objective as long as less than $2^{n/2-5}$ n-bit blocks are processed by the block cipher with a given key. For small block sizes, e.g. 64 bits, and high throughput applications, e.g. securing gigabit network, this may be an issue, e.g., when the key renewal rate is lower than what is required by the data throughput.

Note 3-QuantumThreat.

There exist algorithms leveraging quantum computers, like the Grover algorithm, that may speed up attacks against block ciphers. Such attacks may be performed "retroactively": the ciphertext can be stored now and decrypted later when a quantum computer becomes available. However, the speed up w.r.t. exhaustive search of a k-bit key allowed by a (parallel) implementation of Grover's algorithm is upper bounded by the circuit depth of the computations (measured as an equivalent number of sequential AES computations) and this considerably limits the achievable speed-up as compared with the naive $2^{k/2}$ speed-up factor one would get if ignoring circuit depth limitations. Since one can consider the former speed up limitation effect sufficient in order for AES-128 to remain resistant against quantum attacks beyond the emergence of cryptographically relevant quantum computers (CRQCs), AES-128 remains agreed even in contexts where quantum security is sought. However, the quantum security level of AES-128 could arguably drop under 128 bits: in contexts where quantum security is sought, the use of AES-192 or AES-256 is recommended. ~~[Given the limited validity period of TripleDES-3K, one does not want to take it into consideration here]~~ ~~In contexts where resistance against attacks leveraging quantum computers is required, it is recommended not to use block ciphers with key size smaller than 192 bits.~~

2.2 Stream Ciphers

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

While the present version of this document provides no agreed dedicated stream cipher and therefore no table of agreed stream cipher primitives is introduced in this section, agreed modes of operation of a block cipher such as the counter mode provide, when applied to an agreed block cipher, an agreed stream cipher mechanism.

2.3 Hash Functions

Cryptographic hash functions are keyless functions that take a bit string of arbitrary length as input and produce a fixed-length hash value. General purpose hash functions are required to satisfy many security requirements that include collision resistance, pre-image resistance, and second pre-image resistance.

While SHA-1 is not an agreed hash function, one particular message authentication code whose construction is based upon SHA-1, namely HMAC-SHA-1, is accepted as an admissible scheme.

Agreed Hash Functions

Primitive	Parameters' sizes (hash length h)	R/A	Notes
SHA-2 [FIPS180-4, ISO10118-3]	$h = 256$ bits (SHA-256)	A	4-QuantumThreat
	$h = 384$ bits (SHA-384)	R	
	$h = 512$ bits (SHA-512)	R	
	$h = 256$ (SHA-512/256)	A	4-QuantumThreat
SHA-3 [FIPS202]	$h = 256$ bits	A	4-QuantumThreat
	$h = 384$ bits	R	
	$h = 512$ bits	R	
~~SHA-2 [FIPS180-4, ISO10118-3]~~	~~$h = 224$ bits (SHA-224)~~	~~L [2025]~~	~~4-QuantumThreat~~
	~~$h = 224$ bits (SHA-512/224)~~	~~L [2025]~~	~~4-QuantumThreat~~

Specific Notes.

Note 4-QuantumThreat. There exist algorithms leveraging quantum computers that may speed up attacks against hash functions.

However, the speed up w.r.t. to classical generic attacks against hash functions is limited by the circuit depth of the computations (measured as an equivalent number of sequential hash computations) and this considerably limits the achievable speed-up as compared with the naive speed-up factor one would get if ignoring circuit depth limitations. Since one can consider the former speed up limitation effect sufficient in order for 256 hash functions to remain resistant against quantum attacks beyond the emergence of cryptographically relevant quantum computers (CRQCs), agreed 256-bit output hash functions remain agreed even in contexts where quantum security is sought. However, the quantum security level of these functions could arguably drop under 128 bits: in contexts where quantum security is sought, the use of 384-bit output hash functions is recommended.

~~In contexts where resistance against attacks leveraging quantum computers is required, it is recommended to not use hash functions with output length smaller than 384 bits.~~

2.4 Extendable-Output Functions (XOF)

An eXtendable-Output Function is a variant of hash function that allows to compute hashes of arbitrary length of a given message. A XOF takes as input a bit string of arbitrary length and the length d of the output and returns a d-bit hash value. Two calls to a XOF for a single message but different output lengths will return two hash values, with one being the prefix of the other. Such functions are generally derived from a hash function by means of a construction that reuses some internal component of the hash function.

Note that when using a XOF to generate short outputs, the security is not only limited by the security level of the XOF but also by the size of the output.

Agreed XOFs

Primitive	R/A	Notes
SHAKE256 [FIPS202]	R	
cSHAKE256 [SP800-185]	R	
SHAKE128 [FIPS202]	A	4-QuantumThreat
cSHAKE128 [SP800-185]	A	4-QuantumThreat

2.5 Secret Sharing

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

Agreed Secret Sharing Schemes.

Primitive	R/A	Notes
Shamir's secret sharing [S79]	R	

3 Symmetric Constructions

Symmetric constructions are built upon symmetric primitives and enable to address a large variety of security objectives. The security of keyed symmetric schemes relies on the confidentiality and integrity of a secret key shared by the legitimate parties. We refer to section 8 for a general discussion of agreed secret key generation mechanisms and guidelines on key management.

3.1 Confidentiality Modes of Operation: Encryption/Decryption Modes

Confidentiality modes are schemes providing an encryption procedure, that transforms plaintext into ciphertext using a secret key, and a decryption procedure, that enables to retrieve the plaintext from the ciphertext and the key. They are based on a block cipher.

Agreed Symmetric Encryption Schemes.

Scheme	R/A	Notes
CTR [SP800-38A, ISO10116]	R*	7-StreamMode
OFB [SP800-38A, ISO10116]	R*	7-StreamMode
CBC [SP800-38A, ISO10116]	R*	8-Padding
CBC-CS (CiphertextStealing) [SP800-38A-Addendum]	R*	
CFB [SP800-38A, ISO10116]	R*	8-Padding

Table 1: R* stands for "recommended under the conditions stated in note Note 6-AddIntegrity, admissible if these conditions are not met"

General Notes.

Note 5-IVType. In order to provide security in a strong sense, the encryption scheme must either be probabilistic and generate a random *initialization vector* to bootstrap encryption, or require an additional input, whose value can only be used once with a given key, i.e. a *nonce*. The specifications of modes of operation describe what is expected (nonce or random IV). Implementations shall follow these specifications, e.g., CBC with a constant or more generally a predictable IV does not follow the CBC specification [SP800-38A] and is not accepted.

Note 6-AddIntegrity. Authenticated encryption schemes provide superior privacy security guarantees than encryption only mechanisms. As a consequence, while all encryption only modes considered above are recommended for the purpose of mode construction, as in 3.5, their standalone use is considered admissible.

Specific Notes.

Note 7-StreamMode. Some symmetric confidentiality schemes operate by masking the plaintext by a keystream generated from the key and IV. When using these so-called stream modes of operation, it is of utmost importance to make sure that no two generated key streams ever overlap. This can be achieved deterministically in some modes, e.g., CTR. It is ensured with overwhelming probability in other modes, e.g., OFB mode, as long as a same IV-key pair is not used (or only with negligible probability) to encrypt two messages under the same key.

Note 8-Padding. Some encryption modes cannot handle naturally the encryption of a last incomplete block. For such modes a specific operation must be performed on the last block. A widespread procedure

consists in *padding* the plaintext with some structured data to ensure its size is a multiple of the blocksize. The verification of the format of the padding during decryption may leak information on the decrypted value in a way that could be used to decrypt any ciphertext. More generally, any verification performed on the format of the decrypted ciphertext may leak information. Developers/evaluators should ensure that the implementation of decryption does not provide an attacker with any such *padding or format oracle*. An alternative to the application of the padding scheme is the use of *ciphertext stealing* [SP800-38A-Addendum].

3.2 Specific Confidentiality Modes: Disk Encryption

General purpose encryption modes enable to encrypt data. However, in order to achieve security in a strong sense, they require an expansion of the data of at least one block due to the addition of an initialization vector or nonce. They may also not provide an efficient way to decrypt a specific part of the ciphertext. In some settings, these two properties are inconvenient. This is notably the case for disk encryption. In such contexts, the use of deterministic encryption modes is tolerated.

Agreed Disk Encryption Schemes.

Scheme	R/A	Notes
XTS [SP800-38E]	R	10-UniqueTweak, 11-AddressTweak
CBC-ESSIV	A	12-UniqueSectorNumber, 13-AddressSectorNumber, 14-CBCMalleability

General Notes.

Note 9-DiskEncStreamMode. Disk encryption modes are by nature deterministic encryption modes, where the initialization vector is derived from the location where the encrypted data is stored. This makes the use of stream modes of operation improper since the ciphertexts corresponding to two different plaintexts stored at the same location would leak the difference between the plaintexts.

Specific Notes.

Note 10-UniqueTweak. The tweak value used to encrypt each complete or incomplete block position in each disk sector shall be unique, i.e. a (set of) disk(s) encrypted under the same key shall never contain two distinct blocks encrypted under the same key and the same tweak value. In the former sentence, blocks are meant as n-bit elements of the alphabet of the block cipher.

Note 11-AddressTweak. Usually, the tweak is derived from the address where the ciphertext is stored. For storage devices or disk drivers which determine the used tweak from logical addresses rather than physical addresses, tweak uniqueness is not ensured a priori and extra care should be applied to assess conformance to 10-UniqueTweak.

Note 12-UniqueSectorNumber. Each disk sector shall have a unique sector number, i.e. a (set of) disk(s) encrypted under the same key shall never contain two distinct sectors encrypted under the same key and the same sector number.

Note 13-AddressSectorNumber. Usually, the sector number is derived from the address where the sector is stored. For storage devices or disk drivers which determine the used sector number from logical addresses rather than physical addresses, sector number's uniqueness is not ensured a priori and extra care should be applied to assess conformance to 12-UniqueSectorNumber.

Note 14-CBCMalleability. CBC-ESSIV does not offer integrity protection. Due to the malleability of CBC decryption, it may be possible for an adversary to manipulate the encrypted disk so that the decryption is meaningful [L13]. If this attack path is a concern, stronger disk encryption mechanisms should be adopted.

3.3 Integrity Modes: Message Authentication Codes

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

Integrity modes can be based on several types of primitives, most notably block ciphers and hash functions. MAC schemes can also be partly built upon universal hash functions.

General Notes.

For reasons pertaining to bandwidth, it is a common practice to truncate the result of a MAC scheme. In order for the resulting scheme to resist guessing attacks, where an adversary tries to forge the MAC of a message by a random guess, the final length t of the MAC should not be too short.

Note 15-MACTruncation96. Unless stated otherwise explicitly for a given mechanism, the truncation of a MAC generated by an agreed MAC mechanism to at least 96 bits is agreed.

Note 16-MACTruncation64. Unless stated otherwise explicitly for a given mechanism, the truncation of a MAC generated by an agreed MAC mechanism to at least 64 bits is considered admissible under the condition that the maximal number of MAC verifications performed for a given key over its lifetime can be bounded by 2^{20}.

These two notes apply in the general case. However, for some specific MAC, it may be necessary to be more restrictive on the matter of MAC truncation. This is notably the case for GMAC, see 25-GMAC-GCM-Bounds.

Agreed MAC Schemes based on a Block Cipher.

Scheme	R/A	Notes
CMAC [SP800-38B, ISO9797-1]	R	
CBC-MAC [ISO9797-1, Algorithm 1, Padding 2]	R	17-FixedInputLength

Specific Notes.

Note 17-FixedInputLength. CBC-MAC is agreed only in contexts where the sizes of all the inputs for which CBC-MAC is computed under the same key are identical. Trivial length extension forgeries can be performed when variable length inputs are allowed.

Agreed MAC Schemes based on a Hash Function.

Scheme	Key size	R/A	Notes
HMAC [RFC2104, ISO9797-2]	$k \geq 250$	R	19-QuantumThreat
	$k \geq 125$	A	
KMAC256 [SP800-185]	$k \geq 250$	R	
KMAC128 [SP800-185]	$k \geq 125$	A	19-QuantumThreat
HMAC-SHA-1 [RFC2104, ISO9797-2, FIPS180-4]	$k \geq 125$	A[2030]	18-HMAC-SHA-1

~~[Remark: it would be consistent to remove the use of 100-bit HMAC keys, but no expiration dates have been set. Is it OK to increase the bound on k]~~

Specific Notes.

Note 18-HMAC-SHA-1. The HMAC construction does not require the collision resistance of the underlying hash function. For the time being, HMAC-SHA-1 is considered as an admissible mechanism, even though SHA-1 is not considered as an acceptable general purpose hash function. It is recommended however to phase out HMAC-SHA-1.

Note 19-QuantumThreat. There exist algorithms leveraging quantum computers, like the Grover algorithm, that may speed up attacks against MAC schemes based on hash functions. In contexts where resistance against attacks leveraging quantum computers is required, it is recommended not to use MAC schemes with key size smaller than 192 bits.

Agreed Universal Hash Function Based MAC Schemes.

Scheme	R/A	Notes
GMAC [SP800-38D]	R	22-GMAC-GCMNonce 23-GMAC-GCMOptions 25-GMAC-GCM-Bounds

3.4 Symmetric Entity Authentication Schemes

These schemes allow an entity to prove its identity to a correspondent by demonstrating its knowledge of a secret. They are interactive by nature, and generally consist in using a MAC scheme or an encryption scheme in a random challenge-response protocol. We provide no list for this type of scheme. Note that even though these schemes may both rely on a MAC scheme, they form a distinct type of schemes, with other security objectives. As a consequence, the same key should not be used by integrity modes and symmetric entity authentication schemes, see 82-KeyUsage.

A necessary condition in order for a scheme based on an encryption scheme or a MAC scheme to be agreed is that the underlying block cipher, resp. MAC be agreed.

Note 20-CollChallenge. The verifier must ensure that a challenge cannot be replayed with non-negligible probability. The challenge could for example be implemented by a random value, whose size is large enough, and that is generated by the verifier.

Agreed challenge sizes.

Challenge size ℓ (in bits)	R/A
$125 \leq \ell$	R
$96 \leq \ell < 125$	A

3.5 Authenticated Encryption

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

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

Agreed Symmetric Authenticated Encryption Schemes.

Scheme	R/A	Notes
Encrypt-then-MAC [BN00]	R	21-DecryptionOrder
CCM [SP800-38C, IS019772]	R	21-DecryptionOrder
GCM [SP800-38D, IS019772]	R	21-DecryptionOrder 22-GMAC-GCMNonce, 23-GMAC-GCMOptions 24-GCMPlaintextLength 25-GMAC-GCM-Bounds
EAX [IS019772]	R	21-DecryptionOrder
~~MAC-then-Encrypt~~ [BN00]	~~L[2025]~~	~~21-DecryptionOrder~~
~~Encrypt-and-MAC~~ [BN00]	~~L[2025]~~	~~21-DecryptionOrder~~

General Notes. Note that the agreed symmetric authenticated encryption schemes are all cryptographic constructions. Some of them use as a primitive only a block cipher, e.g. CCM, GCM, while others rely on a primitive encryption scheme and a primitive message authentication scheme, describing only how they are composed, e.g., Encrypt-then-MAC. In both cases the primitives must be agreed, see 1-AgreedPrimitive.

Note also that 15-MACTruncation96 and 16-MACTruncation64 also apply to symmetric authenticated encryption schemes.

Specific Notes.

Note 21-DecryptionOrder. If the integrity of the ciphertexts is not properly checked before decryption, the developer/evaluator should ensure that the implementation does not instantiate any padding or other error oracle. This means that the implementation does not give away any information on the padding or the format of the plaintext obtained through the decryption of an arbitrary ciphertext. Otherwise, an adversary may be able to decrypt a target ciphertext by exploiting, e.g., error messages, or time side-channel information. Plaintexts should never be sent to consuming applications before their integrity has been checked.

Note 22-GMAC-GCMNonce. The IV must be managed within the security perimeter of the authenticated encryption process. For example, it is crucial to ensure that no adversary can cause the same IV to be reused to protect different (plaintext, associated data) pairs under the same key.

Note 23-GMAC-GCMOptions. Only the following GCM options are agreed: the IV length must be equal to 96 bits; the deterministic IV construction method [SP800-38D, Section 8.2.1] must be used; the MAC length t must be 128 bits.

Note 24-GCMPlaintextLength. By specification, at each invocation of GCM, the length of the plaintext must be at most $2^{32} - 2$ blocks of the underlying block cipher. Checking that this maximal length is not exceeded is required in environments where this might potentially happen.

Note 25-GMAC-GCM-Bounds. As the unforgeability bounds of the agreed GMAC and GCM options of 23-GMAC-GCMOptions are not optimal, the MAC length must be at least 128 bits. Thus no truncation to a final MAC length such as 96 bits must be performed.

3.6 Key Protection

A key protection, a.k.a key wrapping, mechanism enables to securely store or transmit keys, ensuring the confidentiality, integrity and data origin authentication of the key.

Agreed Key Protection Schemes.

Scheme	R/A	Notes
SIV [RFC5297]	R	
AES-Keywrap [SP800-38F, algorithms KW and KWP]	R	

3.7 Key Derivation Functions

A key derivation mechanism enables to derive several keys from a single master key. It generally takes as input three arguments, a secret value K, a (possibly) public value N, and a length n, and generates n bits that can be split into several keys that appear to be independent.

There are a lot of widespread good manners to implement this functionality. The list of agreed key derivation mechanisms given here is as such not meant to be exhaustive.

Agreed Key Derivation Functions.

Scheme	R/A	Notes
NIST SP800-56 ABC [SP800-56A, SP800-56B, SP800-56C]	R	
ANSI-X9.63-KDF [ANSIX9.63]	R	
PBKDF2 [RFC8018]	R	26-PBKDF2-PRF
HKDF [RFC5869]	R	

Specific Notes.

Note 26-PBKDF2-PRF. PBKDF2 is built over a pseudo-random function, that can be instantiated using a MAC generation function. This pseudo-random function shall be an agreed mechanism. In the case where HMAC is used, care has to be taken regarding the key length. Indeed, if the HMAC key is longer than the hash function message block length, the key is hashed. This prehashing in HMAC can lower the effective entropy of the key derived using PBKDF2.

3.8 Password Protection/Password Hashing Mechanisms

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

Agreed Password Hashing Mechanisms.

Scheme	R/A	Notes
Argon2-id [RFC9106]	R	27-Argon Parameters
PBKDF2 [RFC8018]	A	29-Salt, 28-NumberOfIterations

General Notes.

Note 27-Argon Parameters. The following set of parameters of the Argon2-id function is agreed: $t = 1$ iteration, $p = 4$ lanes, $m = 2^{21}$ (2GiB of RAM), 128-bit salt, 256-bit tag size.

Note 28-NumberOfIterations. Password hashing mechanisms offer parameters controlling the complexity of the hashing execution. This allows to increase the work factor of brute-force attacks: a legitimate use of the password hashing requires only one execution, while a brute-force attack requires a large number of executions. Thus the number of iterations of PBKDF2 should be selected as large as possible so that it does not impede the legitimate use.

Note 29-Salt. A salt is a random value, generated when the password is registered, and that is stored along the password verification value. Salting a password hashing mechanism counters precomputation attacks. The length of the salt shall be at least 128 bits.

3.9 Key Combiners

A key combiner mechanism enables to combine keys agreed using several key establishment mechanisms (a cryptographic construction addressed in Section 5.4), or pre-shared keys, in such a way that establishment of the resulting key is secure as long as at least one of the key establishment mechanisms is robust. A key combiner mechanism can be used to implement so-called hybrid key establishment mechanisms, e.g. combining a post-quantum KEM and a classical KEM based on ECDH. It takes as input two or more secret keys, possibly the messages exchanged by the key establishment mechanisms, and outputs a combined key.

Agreed Key Combiners.

Scheme	R/A	Notes
NIST KEM combiners [SP800-227, Section 4.6.2]	R	
CatKDF [ETSI]	R	
CasKDF [ETSI]	R	

Note that these principles are being considered in updates of IKE [RFC9370] and TLS [HybridTLS]: TLS and IKE CHILD SA use the Concatenate-then-KDF principle, whereas IKE SA uses the Cascade-KDF principle. The specification of another context independent key combiner is also currently considered at CFRG [draft-irtf-cfrg-hybrid-kems-10].

4 Asymmetric Atomic Primitives

In asymmetric cryptography, the security is based on the discrepancy between the computational difficulties of two mathematical problems: one is easy whereas the other is hard. That asymmetry is needed in order to ensure the possibility of generating pairs of private and public keys for which it is computationally hard to recover the private key from the public key. There should be no polynomial algorithm allowing to deduce the private key from the public key. More generally, it should not be possible to realize any operation that requires the private key (e.g. the decryption of a ciphertext in the case of an encryption scheme, or the forgery of a valid signature in the case of a signature scheme, etc.) with the sole knowledge of the public key, even if the adversary is given results from private operations that require the private key, where inputs of these private operations are known to or chosen by the adversary.

We start by presenting in this section the asymmetric atomic primitives and corresponding mathematical problems, then describe in the next section the asymmetric constructions that can be achieved by building upon these primitives.

Only the primitives whose parameters satisfy the conditions expressed in the tables below are considered agreed.

4.1 RSA/Integer Factorization

The security of various asymmetric cryptographic schemes, including the well-known RSA schemes, relies on the difficulty of integer factorization in comparison with the easiness of integer multiplication and modular exponentiation.

The RSA primitive consists of a public permutation parametrized by a public key and the private inverse permutation parametrized by the associated private key. This primitive is invoked in various RSA encryption or RSA signature schemes addressed in the next section. These schemes also specify padding and redundancy check conventions, etc. Note that the primitive alone must by no means be considered as a complete encryption or signature scheme since using it without extra conventions would be highly insecure.

Let p and q be prime numbers and $N = pq$ their product, called the modulus. We denote $n = \lfloor \log_2(N) \rfloor + 1$ the bitlength of N. The public key is formed by the modulus N together with an element e, called the public exponent, which is invertible modulo $(p-1)(q-1)$. An inverse of e modulo $\mathrm{lcm}(p-1, q-1)$, denoted by d, is called the private exponent. The private key is formed by this private exponent together with the modulus. The public permutation operates on integers modulo N and consists in the exponentiation of the input to the power e. The private permutation operates on integers modulo N and consists in the exponentiation of the input to the power d.

Agreed RSA primitive sizes.

Primitive	Parameters' sizes	R/A	Notes
RSA	$n \geq 3000$, $\log_2(e) > 16$	A	31-LegacyRSA
	~~$n \geq 1900$, $\log_2(e) > 16$~~	~~L [2025]~~	~~31-LegacyRSA~~

General Notes.

Note 30-QuantumThreat. RSA is vulnerable to low-complexity attacks, of polynomial complexity, in the quantum computation model, such as Shor's algorithm. Such attacks may be performed "retroactively": the ciphertext can be stored now and decrypted later when a quantum computer becomes available. In contexts where resistance against attacks leveraging quantum computers is required, the RSA primitive shall not be used without being combined with a quantum resistant mechanism.

Specific Note.

Note 31-LegacyRSA. The use of modulus of size above 1900 bits, but less than 3000 bits, is not agreed but an acceptability deadline for user/data authentication with this particular algorithm may be set on a national level.

4.2 Discrete Logarithm in Finite Fields

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

There is in principle a variety of choices for the finite field, but the only secure and widely used solution is to pick a prime field $\mathsf{GF}(p)$ where p is a prime number. From now on, we restrict ourselves to this case.

The primitive that relies on the discrete logarithm problem in (the multiplicative group of) $\mathsf{GF}(p)$ can be used in various key exchange, signature, or (hybrid) encryption schemes which are described in the next section. Let g be a generator for a subgroup of order q of the multiplicative group $\mathsf{GF}(p)^\times$. Let r be the largest prime factor of q. The primitive is the exponentiation function of base g in $\mathsf{GF}(p)$ that on input an integer x (typically between 1 and $q-1$) returns $X = g^x$. Depending on how the scheme uses the primitive, x and X may represent (a part of) a private key and the associated public key, or they may represent an ephemeral Diffie-Hellman exponent and the associated public value, etc.

Agreed FF-DLOG Parameters.

Family	Group	R/A	Notes
MODP [RFC3526]	3072-bit MODP Group	A	
	4096-bit MODP Group	A	
	6144-bit MODP Group	A	
	8192-bit MODP Group	A	
	~~2048-bit MODP Group~~	~~L[2025]~~	~~34-Precomputation, 35-LegacyFF-DLOG~~
FFDHE [RFC7919]	3072-bit FFDHE Group	A	
	4096-bit FFDHE Group	A	
	6144-bit FFDHE Group	A	
	8192-bit FFDHE Group	A	
	~~2048-bit FFDHE Group~~	~~L[2025]~~	~~34-Precomputation, 35-LegacyFF-DLOG~~

For all agreed subgroups, we have $r = q = (p-1)/2$.

Agreed FF-DLOG Parameters Generation Method. Beside using agreed parameters, another possibility is to generate fresh groups, e.g. following [FIPS186-4, Appendices A.1.1.2, A.2.3] with an agreed hash function. For FF-DLOG parameters with the size of q smaller than the size of p, used e.g. in the context of digital signature, such a method can be applied.

Scheme	R/A
Generation of (p, q) parameters [FIPS186-4, Appendix A.1.1.2]	A
Generation of g [FIPS186-4, Appendix A.2.3]	A

These methods generate a subgroup of prime order, i.e., q is prime and thus $r = q$. The following parameter sizes are agreed.

Primitive	Parameters' sizes	R/A	Notes
FF-DLOG	$\log_2(p) \geq 3000$, $\log_2(q) \geq 250$	A	35-LegacyFF-DLOG
	~~$\log_2(p) \geq 1900$, $\log_2(q) \geq 200$~~	~~L [2025]~~	~~34-Precomputation, 35-LegacyFF-DLOG~~

General Notes.

Note 32-CorrectSubgroup. It should be ensured that manipulated values have order divisible by r and dividing q. In the case of parameters generated with the agreed method, this boils down to checking that manipulated values have exact order q.

Note 33-QuantumThreat. There exist algorithms, like Shor's algorithm, that break completely the discrete logarithm on multiplicative group over finite fields in the quantum computation model. Such attacks may be performed "retroactively": the communications can be stored now and the discrete logarithm can be broken later when a quantum computer becomes available. In contexts where resistance against attacks leveraging quantum computers is required, the discrete logarithm in finite fields primitive shall not be used without being combined with a quantum resistant mechanism.

Specific Notes.

Note 34-Precomputation. ~~Discrete logarithm algorithms involve a group related precomputation phase, which is the bottleneck in terms of complexity of the attack. As a consequence, for DL modules shared by a lot of users and applications, it is strongly recommended not to use modules of length close to the lower limit of the legacy range.~~

Note 35-LegacyFF-DLOG. The use of modules of size above 1900 bits, but less than 3000 bits, is not agreed but an acceptability deadline for user/data authentication with this particular algorithm may be set on a national level.

4.3 Discrete Logarithm in Elliptic Curves

The difficulty of the discrete logarithm problem can also be considered in the group of rational points of an elliptic curve over a finite field. In these groups, the discrete logarithm problem is also thought to be difficult, in comparison to the easiness of scalar point multiplication.

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

Like in the case of finite fields, only elliptic curves defined over prime fields are agreed.

The following notation is introduced in order to describe the basic set-up for a typical asymmetric scheme using the discrete logarithm problem on an elliptic curve defined over a prime field. Let p be a prime number and $\mathsf{GF}(p)$ the prime field with p elements. Let E be an elliptic curve defined over $\mathsf{GF}(p)$, P a point in $E(\mathsf{GF}(p))$ of order q. Let r be the largest prime factor of q. The primitive associated with the discrete logarithm problem on E is the scalar multiplication: on input an integer x between 1 and $q-1$, it returns the point $Q = [x]P$. The public parameters in cryptographic schemes where the primitive is used are formed by p, E, P, and q. Depending on the cryptographic scheme where the primitive is invoked, x and Q may represent (a part of) a private key and the associated public key, or they may represent an ephemeral Diffie-Hellman private value and the associated public value, etc.

Agreed Elliptic Curve Parameters.

Curve Family	Curve	R/A	Notes
Brainpool [RFC5639]	BrainpoolP256r1	A	
	BrainpoolP384r1	A	
	BrainpoolP512r1	A	
NIST [FIPS186-5, Appendix D.1.2]	NIST P-256	A	40-SpecialP
	NIST P-384	A	
	NIST P-521	A	
Edwards curves [RFC7748]	Curve25519	A	40-SpecialP
	Curve448	A	
FR [JORF]	FRP256v1	A	

General Notes.

Note 36-PointOnCurve. Special precautions should be taken to ensure that manipulated points lie on the curve, i.e. they verify the curve equation.

Note 37-PointInSubgroup. In case a curve with non-prime order is used, it should be ensured that the manipulated points lie in the intended subgroup and have order divisible by r.

Note 38-PrimeOrder. If the subgroup order is chosen to be prime, i.e. $q = r$, and such that r^2 does not divide $\#E(\mathsf{GF}(p))$, the verifications for 37-PointInSubgroup boil down to checking that the manipulated points have exact order r.

Note 39-QuantumThreat. There exist algorithms, like Shor's algorithm, that break completely the discrete logarithm on elliptic curves in the quantum computation model. Such attacks may be performed "retroactively": the communications can be stored now and the discrete logarithm can be broken later when a quantum computer becomes available. In contexts where resistance against attacks leveraging quantum computers is required, the discrete logarithm on elliptic curves primitive shall not be used without being combined with a quantum resistant mechanism.

Specific Notes.

Note 40-SpecialP. The special form of the prime number p used to construct the finite field $\mathsf{GF}(p)$ makes side channel attacks more efficient than with a random prime if insufficient countermeasures against side channel attacks are deployed (and not only because the arithmetic of the underlying finite field is faster).

The mathematical problems covered in subsections 4.1 to 4.3 might become vulnerable to quantum computer assisted cryptanalysis, through attacks using Shor's algorithm. There exist other mathematical problems leading to trapdoor or proof of knowledge constructions and for which no generic efficient solving method is known neither in the classical computation model not in the quantum computation model. The Learning With Errors (LWE) problem on Lattices is such an intractable problem whose applications in cryptography have been studied over the past decade.

4.4 Learning With Errors in (Structured) Lattices

A lattice is a discrete set of points presenting a periodic structure. It is defined as the set of linear combinations with integer coefficients of a set of vectors in a high-dimensional space. Problems such as finding short vectors in a lattice are considered to be hard to solve. The Learning With Errors problem, that is being used in several candidate cryptographic schemes, consists in solving modulo an integer q a noisy linear system of equations in the related lattice. Given a matrix A and a vector $b = A \cdot z + e \mod q$, with e a noise vector, recover z. The difficulty of this problem depends notably on the modulus q, the dimension of the lattice and the distribution of the noise vector e.

In order to improve the performance of LWE based mechanisms, the A matrices may be considered with an additional structure, which enables to speed up the computations and may reduce the size of objects manipulated by the scheme. This gives rise to variants of the LWE problem, such as the Module Learning With Errors (MLWE) problem.

In LWE based cryptographic mechanisms, the lattice on which the scheme operates tends to be built-in in the scheme. As a consequence, we do not specify here standalone agreed LWE parameters. For an agreed LWE based scheme, the agreed status covers the cryptographic mechanism as well as its (M)LWE parameters.

General Notes.

Note 41-Hybridization. LWE or MLWE based cryptographic mechanisms shouldn't be used in a standalone way to provide the intended security functionality, but should be combined with a well established classical cryptomechanism.

In order to provide assurance that the introduction of new asymmetric primitives does not risk regression of the security, due to evolving state of the art of their analysis and secure implementations, it is recommended to continue to use well-tested classical agreed mechanisms in a combined way with new PQC cryptographic resistant mechanisms.

4.5 Hash Function Preimage Resistance

The security of several signature schemes relies on the difficulty of computing preimages of a cryptographic hash function (cf Section 2.3) or XOF (cf Section 2.4), in comparison with the easiness of applying the ~~hash~~ function on a given value.

General Notes.

Note 42-OptionalHybridization. In order to provide additional assurance, hash-based asymmetric cryptographic mechanisms can be combined with a classical cryptographic mechanism to provide the intended security functionality. They may however also be used in a standalone way.

4.6 Other Intractable Problems

There exist other mathematical problems leading to trapdoor or proof of knowledge constructions and for which no generic efficient (possibly quantum) solving method is known in general.

To cite a few:

- code-based cryptography, that is problems relying on the difficulty of decoding a random error correcting code (candidate for quantum resistance);
- multivariate cryptography (candidate for quantum resistance).

5 Asymmetric Constructions

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

The security of asymmetric schemes relies on the security of a mathematical problem, and can also rely on the security of a symmetric primitive or scheme. We recall that in the general case, in order for

a cryptographic construction to be agreed, it has to be based on agreed primitives. In particular, the requirements on parameter sizes established in the previous section also apply for the schemes in this section.

The security of keyed asymmetric schemes relies on the confidentiality and integrity of the private key and the integrity and data origin authentication of the public key. We refer to section 8 for a general discussion of agreed asymmetric key-pair generation mechanisms and guidelines on key management. For each keyed asymmetric scheme considered in this section, specific aspects of the key-pair generation are addressed in the same subsection as the rest of the scheme.

5.1 Asymmetric Encryption Scheme

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

Agreed Asymmetric Encryption Schemes.

Primitive	Scheme	R/A	Notes
RSA	OAEP (PKCS#1v2.1) [RFC8017, PKCS1]	A	45-OAEP-PaddingAttack, 46-QuantumThreat
RSA	PKCS#1v1.5 [RFC8017, PKCS1]	A	44-PaddingAttack, 46-QuantumThreat

General Notes.

Note 43-RandomPadding. The asymmetric encryption schemes use a randomized padding that shall be generated by an agreed random bit generator, see Section 7.

Specific Notes.

Note 44-PaddingAttack. In case a padding oracle is available, the RSA-PCKS#1v1.5 scheme is vulnerable to efficient attacks.

Note 45-OAEP-PaddingAttack. In case the OAEP decryption procedure is not correcty implemented, that is to say, the checks performed by EME-OAEP decoding are not performed in the specified order, RSA OAEP may also be vulnerable to oracle attacks.

Note 46-QuantumThreat. There exist algorithms leveraging quantum computers, like Shor's algorithm, that break completely the asymmetric primitives these cryptographic constructions are based on. Such attacks may be performed "retroactively": the ciphertext can be stored now and decrypted later when a quantum computer becomes available. In contexts where resistance against attacks leveraging quantum computers is required, these cryptographic mechanisms shall not be used without being combined with a quantum resistant mechanism.

5.2 Digital Signature

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

Agreed Digital Signature Schemes.

Primitive	Scheme	R/A	Notes
RSA	PSS (PKCS#1v2.1) [RFC8017, PKCS1, ISO9796-2]	A	52-QuantumThreat
FF-DLOG	KCDSA [ISO14888-3]	A	50-DSARandom, 52-QuantumThreat
	Schnorr [ISO14888-3]	A	
	DSA [FIPS186-5, ISO14888-3]	A	
EC-DLOG	ECKCDSA [ISO14888-3]	A	50-DSARandom,52-QuantumThreat
	ECDSA [FIPS186-5, ISO14888-3]	A	
	ECGDSA [TR-03111]	A	
	ECSchnorr [ISO14888-3]	A	
	Determistic ECDSA [FIPS186-5]	A	52-QuantumThreat
	EdDSA [RFC8032]	A	52-QuantumThreat,51-~~DeterministicSignatures~~
LWE	ML-DSA [FIPS204]	R	53-Hybridization, 58-ML-DSA Parameters
HASH	XMSS [SP800-208]	R	55-OptionalHybridization, 56-StateManagement, 57-XMSS-LMS-Parameters
	LMS [SP800-208]	R	
	SLH-DSA [FIPS205]	R	54-SLH-DSAParameters, 55-OptionalHybridization
RSA	PKCS#1v1.5 [RFC8017, PKCS1, ISO9796-2]	A	49-PKCSFormatCheck,52-QuantumThreat

General Notes.

Note 47-Hash. The scheme is agreed provided the underlying hash function is.

Note 48-DifficultProblem. The scheme is agreed provided the underlying mathematical problem uses agreed parameters.

Specific Notes.

Note 49-PKCSFormatCheck. Format checks should be carefully implemented to avoid attacks à la Bleichenbacher.

Note 50-DSARandom. In DSA and its elliptic curve variants, the signature generation procedure generates a random value. Leakage of the random per-signature values used during signature generation poses risks to the confidentiality of the associated long-term keys. Such leakage should therefore be avoided. This pertains in principle both to statistical leakage through biases in the random number generator used and to leakages on the value of particular random bits as may be obtained by an attacker, e.g. through side-channel analysis.

It is therefore recommended to use a strong random number generator with strong cryptographic post-processing, enhanced backward security, and regular reseeding from a true random source for such applications, see Section 7.3.

Note 51-~~DeterministicSignatures~~. ~~*[proposal to take into account the attacks mentionned by J. Mittmann, to be dis*~~
~~In EdDSA, the secret per-signature value is generated pseudo-randomly, from the secret key and the message. While an implementation of this mechanism does not require a random generator, it is susceptible to fault attacks targeting the hash of the message once the secret value has been computed.~~

Note 52-QuantumThreat. There exist algorithms leveraging quantum computers, like Shor's algorithm, that break completely the asymmetric primitives these cryptographic constructions are based on. In contexts where resistance against attacks leveraging quantum computers is required, these cryptographic mechanisms shall not be used without being combined with a quantum resistant mechanism.

Note 53-Hybridization. These cryptographic mechanisms are based on novel asymmetric primitives. To provide assurance against regression of robustness, they shouldn't be used in a standalone way to provide the intended security functionality, but should be combined with a classicaly secure cryptographic mechanism.

For digital signatures, hybridization can consist in concatenating signatures from different schemes, the verification function accepting if and only if all signatures are correct.

Note 54-SLH-DSAParameters. The parameters defined in [FIPS205, Table 2] with security level 3 and 5 are agreed. This digital signature schemes does not require the collision resistance of the inner hash functions or XOFs used to compute hash trees. As a consequence, it is are agreed even when using hash functions with outputs of length 192 bits derived from agreed hash functions.

Note 55-OptionalHybridization. In order to provide additional assurance, hash-based asymmetric cryptographic mechanisms can be combined with a classical cryptographic mechanism to provide the intended security functionality. They may however also be used in a standalone way.

Note 56-StateManagement. The state of the signature algorithm is a critical data that must be protected in integrity and against replay attacks.

Note 57-XMSS-LMS-Parameters. These digital signature schemes do not require the collision resistance of the inner hash functions or XOFs used to compute hash trees. As a consequence, they are agreed even when using hash functions with outputs of length 192 bits derived from agreed hash functions.

Note 58-ML-DSA Parameters. It is recommended to use the highest possible standardised parameter size, either ML-DSA-87 or ML-DSA-65.

5.3 Asymmetric Entity Authentication Schemes

These schemes allow an entity to prove its identity to a correspondent by demonstrating its knowledge of a private key. They are interactive by nature, and generally consist in using a signature scheme in a random challenge response protocol. We provide no list for this type of scheme. Note that even though these schemes may rely on a signature scheme, they form a distinct type of schemes, with other security objectives. As a consequence, the same key should not be used by a signature scheme and an asymmetric entity authentication scheme, see 82-KeyUsage.

As for symmetric entity authentication schemes, the challenge should verify some properties, see 20-CollChallenge.

5.4 Key Establishment and Key Encapsulation

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

The most widely used two-party key establishment scheme has been proposed by Diffie and Hellman and relies on the discrete logarithm problem (instantiated in any suitable group). It proceeds as follows for two users:

1. Both users agree on a group G and a generator g.
2. Each user picks a random value r_i, $i \in \{1, 2\}$, and sends g^{r_i} to the other user.
3. Both users can then compute $g^{r_1 r_2}$ from their own random value and the element sent by the other user.

An alternative framework, the Key Encapsulation Method, enables for a party to share a symmetric key with another party. The recipient starts by generating a key pair (sk, pk). The sender can then *encapsulate* under the given public key pk, which generates a secret key k and a ciphertext c. The recipient can finally decapsulate the secret key k from the ciphertext, using her private key sk.

It should be noted that in such a basic version the protocol is among other issues vulnerable to man-in-the-middle attacks. In particular, additional steps should be performed and additional data should be exchanged to ensure authentication of the users and of the key establishment messages.

Agreed Key Establishment and Key Encapsulation Schemes.

Primitive	Scheme	R/A	Notes
FF-DLOG	DH [SP800-56A, ISO11770-3]	A	60-KE-Auth, 61-DHSubgroupAttacks 62-QuantumThreat
	DLIES-KEM [ISO18033-2]	A	
EC-DLOG	EC-DH [SP800-56A, ISO11770-3]	A	60-KE-Auth, 61-DHSubgroupAttacks, 62-QuantumThreat
	ECIES-KEM [ISO18033-2]	A	
	X25519, X448 [RFC7748]	A	60-KE-Auth, 62-QuantumThreat
LWE	ML-KEM [FIPS203]	R	60-KE-Auth, 63-Hybridization, 64-ML-KEM Parameters
	FrodoKEM [FRODO-KEM]	R	60-KE-Auth, 63-Hybridization, 65-FrodoKEM Parameters

General Notes.

Note 59-DifficultProblem. The scheme is agreed provided the underlying mathematical problem uses agreed parameters.

Specific Notes.

Note 60-KE-Auth. Unauthenticated key establishment that may fall to man-in-the-middle attacks. In order to ensure security, it is necessary to authenticate the other party, the data exchanged during the key establishment scheme, such as public points, public keys/ciphertexts, and identities. This authentication requires long-term secrets.

Note 61-DHSubgroupAttacks. Whether the Diffie–Hellman protocol is defined over the multiplicative group of a finite field, or the group of rational points of an elliptic curve, it should be ensured that the manipulated values lie in the intended subgroup and have a large enough order as specified by 32-CorrectSubgroup in the finite field case and 36-PointOnCurve, 37-PointInSubgroup in the elliptic curve case.

Note 62-QuantumThreat. There exist algorithms leveraging quantum computers, like Shor's algorithm, that break completely the asymmetric primitives these cryptographic constructions are based on. Such attacks may be performed "retroactively": the ciphertext can be stored now and decrypted later when a quantum computer becomes available. In contexts where resistance against attacks leveraging quantum computers is required, these cryptographic mechanisms shall not be used without being combined with a quantum resistant mechanism.

Note 63-Hybridization. (M)LWE based cryptographic mechanisms shouldn't be used in a standalone way to provide the intended security functionality, but should be combined with a classical cryptomechanism.

Note 64-ML-KEM Parameters. It is recommended to use the highest possible standardised parameter size, either ML-KEM-1024 or ML-KEM-768.

Note 65-FrodoKEM Parameters. It is recommended to use the highest possible standardised parameter size, either FrodoKEM-1344 or FrodoKEM-976.

6 Cryptographic Protocols

6.1 TLS

Transport Layer Security (TLS) [RFC5246, RFC8446], formerly also known as Secure Socket Layer (SSL), is a protocol to protect the communication over the Internet, e.g. connection via HTTP (HTTPS). TLS enables to set up and use secure channels between a *Client* and a *Server*.

The *Handshake Protocol* enables to set up secure channels, that is to say to choose protocol version, a common set of cryptographic mechanisms, the *ciphersuite*, establish shared keys between the Client and the Server, authenticate the Server to the Client, and optionnaly the Client to the Server. The result of the execution of this protocol is a set of session data.

The *Record Protocol* uses the result of the established session data to protect the confidentiality and integrity of subsequent communications, e.g., application data or refreshment of session data.

Agreed protocol version

TLS protocol version	R/A
TLSv1.3 [RFC8446]	R
TLSv1.2 [RFC5246]	A

Note 66-TLSversion. In order to avoid protocol downgrade attacks, SSLv2 must not be supported.

Agreed TLS Cipher Suites It is recommended to use a ciphersuite offering perfect forward secrecy (PFS). Following 1-AgreedPrimitive, every component of an agreed cipher suite shall be chosen as to be an agreed mechanism.

TLS code	Name	R/A	Notes
TLS v1.3 Cipher Suites			
0x1302	TLS_AES_256_GCM_SHA384	R	
0x1301	TLS_AES_128_GCM_SHA256	A	
0x1304	TLS_AES_128_CCM_SHA256	A	
TLS v1.3 Supported Groups			
0x0017 - 0x0019	secp<256,384,521>r1	A	
0x001D - 0x001E	x25519, x448	A	
0x001F - 0x0021	brainpoolP<256,384,512>r1tls13	A	
0x11EB	SecP256r1MLKEM768	R	
0x11EC	X25519MLKEM768	R	
0x11ED	SecP384r1MLKEM1024	R	
TLS v1.2 Cipher Suite			
0xC02C	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	A	
0xC02B	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	A	
0xC0AD	TLS_ECDHE_ECDSA_WITH_AES_256_CCM	A	
0xC0AC	TLS_ECDHE_ECDSA_WITH_AES_128_CCM	A	
0xC024	~~TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384~~	L[2025]	67-TLSEncryptThenMAC
0xC023	~~TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256~~	L[2025]	67-TLSEncryptThenMAC
0xC028	~~TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384~~	L[2025]	67-TLSEncryptThenMAC
0xC027	~~TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256~~	L[2025]	67-TLSEncryptThenMAC
0xC030	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	A	
0xC02F	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	A	
0x009F	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	A	68-TLSDHE
0x009E	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	A	68-TLSDHE
0xC09F	TLS_DHE_RSA_WITH_AES_256_CCM	A	68-TLSDHE
0xC09E	TLS_DHE_RSA_WITH_AES_128_CCM	A	68-TLSDHE
0x006B	~~TLS_DHE_RSA_WITH_AES_256_CBC_SHA256~~	L[2025]	68-TLSDHE,67-TLSEncryptThenMAC
0x0067	~~TLS_DHE_RSA_WITH_AES_128_CBC_SHA256~~	L[2025]	68-TLSDHE,67-TLSEncryptThenMAC
0x009D	TLS_RSA_WITH_AES_256_GCM_SHA384	A	69-TLSRSA
0x009C	TLS_RSA_WITH_AES_128_GCM_SHA256	A	69-TLSRSA
0xC09D	TLS_RSA_WITH_AES_256_CCM	A	69-TLSRSA
0xC09C	TLS_RSA_WITH_AES_128_CCM	A	69-TLSRSA
0x003D	~~TLS_RSA_WITH_AES_256_CBC_SHA256~~	L[2025]	69-TLSRSA,67-TLSEncryptThenMAC
0x003C	~~TLS_RSA_WITH_AES_128_CBC_SHA256~~	L[2025]	69-TLSRSA,67-TLSEncryptThenMAC

Specific Notes.

Note 67-TLSEncryptThenMAC. The use of TLS ciphersuites whose encryption mechanisms is based on CBC should be used together with the `encrypt_then_mac` extension.

Note 68-TLSDHE. In TLS 1.2, when DHE key exchange is used, the group parameters are imposed by the server to the client, and validation of these parameters by the client may be problematic. Use of ECDHE key exchange, for which the group parameters are negotiated in the Handshake Protocol should be preferred.

Note 69-TLSRSA. RSA key exchange does not offer PFS.

7 Random Generator

7.1 True Random Source

A true random source is a probabilistic process from which random bits can be extracted. It is typically very difficult to assess the quality of the output of a random source. There are basically two approaches :

- **Perform statistical tests on the output of the source.** This approach is black blox, no knowledge about the source is required to conduct the tests. It suffers from two main drawbacks. First, the statistical tests are generic and may only be used to detect specific shortcomings of the source relative to an ideal random source. Second, it does not provide any assurance on the distribution of the output of the random source. Note that despite these shortcomings, statistical tests are useful to detect unintentional failure of the random source.

- **Model the probabilistic process of the source.** This approach requires a deep understanding of the random source design, and tries to assess the quality of the source from the study of a theoretical model of the source. This approach provides better assurance but requires a high level of expertise in several domains, e.g., physics and statistics, to be carried out. Some aspects like the degree of correspondence between the random source and its model may be difficult to evaluate.

Note 70-NoDirectRandomSource. Since true random sources may produce outputs with statistical defects, their use should be limited to providing (re)seed material or additional input for deterministic random bit generators. The direct use of a pure true random generator is not agreed.

7.2 Deterministic Random Bit Generator

In order to lower the level of assurance expected from a random source, its output shall be processed by a Deterministic Random Generator (DRG). This is a (deterministic) cryptographic construction, built around an internal state, that can be seeded and refreshed by output of random sources, and from which (pseudo-)random bits can be extracted.

Agreed Deterministic Random Bit Generator.

Scheme	R/A	Notes
HMAC_DRBG [SP800-90A, ISO18031]	R	
Hash_DRBG [SP800-90A, ISO18031]	R	
CTR_DRBG [SP800-90A, ISO18031]	R	74-QuantumThreat

General Notes.

Note 71-DRG-Seeding. The security of DRG derives from a proper seeding of the internal state of the construction from a random source. The required min-entropy for the (re)seeding operation of the DRG mandated by its specification shall be respected. Furthermore the min-entropy of the seed shall be at least 125.

Note 72-Quantum-Threat. In contexts where resistance against attacks leveraging quantum computers is required, it is recommended that the min-entropy of the seed is at least 188 bits.

Note 73-BacktrackingResistance. In systems which aim at providing perfect forward secrecy (PFS), an attacker who has recovered the current state of the random number generator which was used in previous key exchanges to produce ephemeral keys will be able to break the PFS property if it is possible to practically compute past DRG outputs from the present state of the DRG. Therefore, only DRG which do not allow such backwards computation should be used.

Specific Notes.

Note 74-QuantumThreat. There exist algorithms leveraging quantum computers, like the Grover algorithm, that may speed up attacks against block ciphers. In contexts where resistance against attacks leveraging quantum computers is required, it is recommended not to use CTR_DRBG with block ciphers with key size smaller than 192 bits.

7.3 Random Number Generator with Specific Distribution

In asymmetric cryptographic mechanisms, the need to generate integers following specific distributions arises frequently. In such cases, a random bit generator cannot be used directly, since it does not directly offer the adequate distribution. As a consequence, random generators offering specific output distributions need to be implemented from random bit generators.

Agreed "Mod q uniform distribution" generator Some mechanisms require the use of random numbers generated uniformly at random in an interval of the form $[0, q-1]$, where q is not a power of 2. It is notably the case when an element needs to be drawn at random in a group of prime power.

Scheme	R/A	Notes
"Testing" technique [FIPS186-5, Appendix B.1.2]	R	
"Extra random" technique [FIPS186-5, Appendix B.1.1]	R	

General notes.

Note 75-RandomModularReduction. The integer generation method that consists in using the underlying random bit generator to draw an integer uniformly at random in a range of length 2^ℓ where ℓ is the ceil or the floor of $\log_2(q)$, possibly applying a reduction $\mod q$ to the result, introduces biases in the generation that may lead to attacks.

The "testing" technique ensures uniform generation $\mod q$ at the cost of the use of possibly extra, variable amount of randomness. The "extra random" technique makes the biases negligible at the cost of a fixed, small amount of extra randomness.

Prime generation. The generation of asymmetric parameters and RSA keys requires to generate random prime numbers. The generation of random primes adopts the following strategy: initially an integer of the appropriate size is drawn at random. Then, it is tested whether it satisfies some properties and whether it is prime. As long as these tests fail, the candidate integer is updated and tested anew. The choice of the updating function offers a tradeoff between the amount of extra randomness needed and the closeness of the generated prime distribution to the uniform distribution of primes. The primality test can be either a pseudo-prime test, or a provable-prime test.

Agreed Prime Generation Methods.

Scheme	R/A	Notes
Prime generation by rejection sampling (Method 1) Appendix B.1	R	
Prime generation by more efficient rejection sampling (Method 2) Appendix B.2	R	
Prime generation based on auxiliary primes [FIPS186-5, Appendix C.9]	R	

These methods are not vulnerable to the ROCA attack [ROCA].

Agreed Primality Tests.

Scheme	R/A	Notes
MillerRabin [HAC, Algorithm 4.24]	R	76-ProbablePrime

Note 76-ProbablePrime. In case the primality test does not prove the primality of the tested number, the probability that this number is composite should be lower that 2^{-125}.

Following[FIPS186-5, Appendix F.1], the following table gives the number t of Miller-Rabin test iterations that is necessary to reach this probability when testing a random k-bit odd number. In particular, 6 iterations are needed when testing 1024-bit prime candidates, and 4 are required for 1536-bit prime candidates. Note that every of these iterations operates on a randomly selected basis a.

Table 2: Number t of Miller-Rabin test iterations according to the bitlength k of randomly generated odd input to ensure to ensure that the probability of the input passing the test despite being composite is lower that 2^{-125}.

t	k
6	$950 \leq k < 1041$
5	$1041 \leq k < 1297$
4	$1297 \leq k < 1729$
3	$1729 \leq k < 2626$
2	$2626 \leq k < 5701$
1	$5701 \leq k$

RSA key generation. The generation of RSA keys consists in generating a pair of random primes satisfying a set of constraints to ensure the size of the key, the consistency of the public/private key pair, and to avoid weak keys.

Scheme	R/A	Notes
RSA key-pair generation B.3	R	

General notes.

RSA key pairs generation relies on a random prime generator. Furthermore, additional conditions have to be satisfied.

Note 77-RSAKeyGen. The prime numbers p and q should be two randomly generated primes of same length, whose product have the given modulus bitlength. The two primes should not be too close to avoid factorization attacks exploiting a short distance between the two factors. One should have

$$|p - q| \geq 2^{\frac{n}{2}-100}.$$

Note 78-SmallD. The size of d should be sufficiently large, that is to say $d > 2^{n/2}$, where n denotes the bitlength of the modulus. Note that this is guaranteed for a small e.

Note that these conditions are satisfied by B.3.

8 Key Management

In the general case, the security of cryptographic mechanisms relies on the confidentiality, integrity, and/or authenticity of some *keys*. For the mechanism to be secure, it is crucial that the keys are not compromised/altered by an adversary. Agreed cryptographic mechanisms are deemed to be robust. As a consequence, using the mechanism does not endanger the confidentiality of the key it relies on. However, when considering/evaluating a product implementing cryptographic mechanisms, one has to consider every way the product manipulates key material and every way an adversary may target the product in order the ensure that she cannot recover said keys. The guiding principle can be expressed as follows.

Note 79-KeyManagement. The management of keys by the product should not enable a potential attacker to recover any information about secret and private keys used to protect user information, nor to alter or inject public keys used to protect identities.

This principle impacts all steps of the key life cycle. It has to be noted that whether a product satisfies the principle is not straightforward and may depend on the capabilities of the provisioned adversary. Below we refine 79-KeyManagement for some key life cycle steps.

Symmetric versus asymmetric techniques. Symmetric cryptosystems usually restrict the ability to read or authenticate communications to a closed group of users. Symmetric keys must be distributed to these users and it is of paramount importance that nobody outside the closed user group be privy to

those keys; protection of the key distribution channel for authenticity and integrity is also very important. With asymmetric cryptosystems, on the other hand, keys are split into a public and a private part: the private part can be generated locally and needs to be protected for confidentiality at all costs. The public part can be sent over a nonconfidential channel, but must reliably be protected for authenticity and integrity.

8.1 Key Generation

In order for an adversary to have no a priori knowledge of the keys used by a cryptographic mechanism, they must be unpredictable. It is required that the keys are long enough and that the output distribution of the process used to generate them cannot be distinguished from the uniform distribution from an adversary point of view.

Agreed key generation methods. We give here the agreed key generation methods for the generic keyed cryptographic mechanisms. Unless stated otherwise, the keys used for the agreed cryptographic mechanisms of the preceding sections shall be obtained by truncating a bit sequence output by an agreed generation method to the size of the key of the mechanism.

Method	Notes
Agreed random bit generator	
Agreed key establishment mechanism	80-KeyEstablishment,81-KeyGenerationSeed
Agreed key derivation function	81-KeyGenerationSeed

Note 80-KeyEstablishment. A key established through key establishment should not be used directly but first postprocessed by an agreed key derivation function.

Note 81-KeyGenerationSeed. Some key generation mechanisms rely on preexisting secrets, e.g., ephemeral exponents in the case of a Diffie-Hellman key establishment protocol, or a master secret in the case of the generation of TLS Record Protocol keys. The entropy of the preexisting secrets shall be at least 125 bits.

For mechanisms having specific needs in terms of key generation, e.g. distribution of keys, they are specified along the mechanisms in the previous sections. In these cases, a specific key generation procedure using a random bit generator as a black box is usually defined.

Key Usage.

Note 82-KeyUsage. A key must not be used with different mechanisms.

Using a key with several mechanisms, to guarantee, e.g., confidentiality, integrity and authentication, is a source of errors and may also open attack paths exploiting the various contexts of usage of the key. Note that the usage is to be understood in terms of achieved security objective, and not restricted in terms of cryptographic mechanisms: for example, a message digital signature context and an asymmetric authentication scheme may both make use of the same digital signature scheme, but the key pairs used in the two contexts must be different.

8.2 Key Storage and Transport

Note 83-KeyStorageAndTransport. When transmitted or stored on a non-trusted medium, a key shall be protected in confidentiality and integrity with an agreed key protection mechanism, as described in Section 3.6.

8.3 Key Use

Note 84-TrustedPlatform. A key shall only be used to perform computations on a trusted platform. Confidentiality and integrity shall be ensured for secret and private keys, integrity and authenticity shall be ensured for public keys.

Note 85-Distribution. The distribution of secret and private keys shall be limited to the trusted environment that makes an effective use of the key.

8.4 Key Destruction

Note 86-KeyDestruction. At the end of the key life-cycle, the key shall be securely erased from the trusted platform where it was used.

In particular, some cryptographic schemes use so-called ephemeral keys. In the context of Diffie-Hellman key establishment schemes, these ephemeral keys are exchanged by the parties and are used to derive a shared secret. Following this derivation, the ephemeral keys are no longer useful and should be securely erased.

The erasure process must be adapted to the environment and take into account memory remanence issues.

9 Person Authentication

In the general case, authentication, where one entity proves its identity to another, is ensured by using cryptographic mechanisms. However, the case where one person identifies itself to an information system is peculiar in several ways. First, the person cannot directly make use of cryptographic mechanisms. Second, the authentication procedure is more prone to be replayed, for instance if it relies on a long-termed password. Third, the entropy of the identification data, as a password, may be substandard regarding what would be expected from a cryptographic system. Thus, such person authentication procedures can only be performed locally on a trusted platform (e.g., entering a PIN), or through a trusted channel (e.g., providing a cookie to a website). Furthermore, it must require an interaction with the system: if identification verification data can be extracted from the system, an attacker may retrieve from it the identification data by use of brute-force. In this document, we limit ourselves to considering password and PIN-based person authentication solutions.

We express limits that shall be enforced by the person authentication system on the authentication procedure, in order to make it difficult for an unauthorized person to authenticate in place of a legitimate user of the system. We consider two cases. In the first case, the system can enforce a hard limit on the number of trials the authenticating person is allowed to perform for entering the correct information. In the second case, such limitation is impractical but the system can limit the rate at which the authenticating person can submit to the authentication procedure.

Limited number of trials. This case is usually implemented by cryptographic resources, e.g., smartcards and HSM, in order to unlock the access to operations involving keys securely stored on the resource. After a given number of erroneous trials, authentication becomes impossible: the person account can no longer be unlocked without resorting to an administrative procedure. In such a case, it is possible to upper bound the probability of false acceptance, i.e., the event that a person that does not know the identification data can succeed in authenticating itself by random trials. It is recommended to use a maximal false acceptance probability of 5×10^{-6}, corresponding to at most 5 trials of a uniformly distributed 6-digit PIN, with digits 0-9. A maximal false acceptance probability of 5×10^{-4}, at most 5 trials of a 4-digit PIN, is acceptable in legacy applications.

Time limited number of trials. This case is usually implemented when it is not practical to enforce a strict account locking mechanism in case of repeated authentication errors. It provides the inferior solution of limiting the amount of authentication trials by unit of time. An example of such limitating mechanism is to double the delay after every unsuccessful authentication attempt before a new attempt

is possible. For the time being, this document does not express any requirement for this case.

A Glossary

Authentication: provision of assurance of the claimed identity of an entity.

Confidentiality: property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

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

Asymmetric key pair: pair of related keys where the private key defines a private transformation and the public key defines a public transformation.

Cryptographic atomic primitive: cryptographic mechanism that is not considered as a cryptographic construction.

Cryptographic construction: cryptographic mechanism built upon (an)other cryptographic mechanism(s). See cryptographic primitive.

Cryptographic function: sequence of instructions that associate deterministically to a set of inputs a corresponding output.

Cryptographic mechanism: general term designating a security-related procedure using cryptography.

Cryptographic primitive: cryptographic mechanism used to build a higher level cryptographic mechanism. See cryptographic construction.

Cryptographic procedure: generalization of cryptographic function that may additionaly make use of random values to compute an output associated to its input.

Cryptographic protocol: protocol which performs a security-related function using cryptography. Can be considered as an interactive cryptographic scheme.

Cryptographic scheme: a distributed cryptographic algorithm, involving several parties, usually sharing related key material, that achieves some security objective.

Data origin authentication: confirmation that the source of data received is as claimed.

Data integrity: property that data has not been altered or destroyed in an unauthorized manner.

Key: sequence of symbols that controls the operation of a cryptographic procedure.

Key management: administration and use of the generation, registration, certification, deregistration, distribution, installation, storage, archiving, revocation, derivation and destruction of keying material in accordance with a security policy.

Non-repudiation: ability to prove the occurrence of a claimed event or action and its originating entities.

Nonce: Value that may not be used more than once.

Secret key: key used with symmetric cryptographic techniques by a specified set of entities.

Security objective: statement of an intent to counter identified threats and/or satisfy identified organization security policies and/or assumptions. Examples of security objectives are confidentiality, data integrity and data origin authentication.

B Prime and RSA key-pair generation

B.1 Prime generation by rejection sampling (Method 1)

Input: an interval $I = [a, b] \cap \mathbb{N}$, in which a prime is to be found, an optional test `Test` encoding additional properties of the prime to generate, and a primality test `TestPrime`.

1. Choose a number p according to the uniform distribution on I.
2. If p is not odd, return to step 1.
3. If $\texttt{Test}(p) = \texttt{False}$, return to step 1.
4. If $\texttt{TestPrime}(p) = \texttt{False}$, return to step 1.
5. Output p.

B.2 Prime generation by more efficient rejection sampling (Method 2)

In the following, we denote Π_B the product of the primes less than B.

Input: an interval $I = [a, b] \cap \mathbb{N}$ in which a prime is to be found, a small natural number B satisfying $\Pi_B \ll b-a$, an optional test `Test` encoding additional properties of the prime to generate, and a primality test `TestPrime`

1. Choose randomly r according to the uniform distribution on $[1, \Pi_B[$.
2. If $\gcd(r, \Pi_B) \neq 1$, return to step 1.
3. Randomly choose $k \in \mathbb{N}$ subject to the constraint that $p := k\Pi_B + r \in I$. Hence, k is to be selected according to the uniform distribution on $[\lceil (a-r)/\Pi_B \rceil, \lfloor (b-r)/\Pi_B \rfloor]$.
4. If $\texttt{Test}(p) = \texttt{False}$, return to step 3.
5. If $\texttt{TestPrime}(p) = \texttt{False}$, return to step 3.
6. Output p.

B.3 RSA key-pair generation

Input: the bitlength n of the output, and e the public exponent

1. If e is even or $e \leq 2^{16}$, output `FAILURE`.
2. Using an agreed random prime generation method, generate a random prime p in interval $[\frac{1}{\sqrt{2}}2^{n/2}, 2^{n/2}]$, so that $\texttt{Test}(p) : \gcd(p-1, e) = 1$.
3. Using an agreed random prime generation method, generate a random prime q in interval $[\frac{1}{\sqrt{2}}2^{n/2}, 2^{n/2}]$, so that $\texttt{Test}(q) : \gcd(q-1, e) = 1$ and $|p-q| \geq 2^{n/2-100}$.
4. Compute $d = e^{-1} \mod \operatorname{lcm}(p-1, q-1)$.
5. If $d \leq 2^{n/2}$, return to step 1.
6. Output p, q.

C Update process for Agreed Cryptographic Mechanisms

The following process has been adopted by the European Cybersecurity Certification Group Sub-group on Cryptography for the update of Agreed Cryptographic Mechanisms.

Conditions for submission of a new mechanism. Following conditions apply to a mechanism submitted to inclusion into ACM:

- it is standardised;
- it is stable for at least two years;
- it has a generic purpose and several applications/use cases;
- its security level should be at least of 125 bits (the mechanism will enter at Recommended level);
- it has been submitted to peer reviews, security proofs or security analysis by academics, side-channel analysis, tentative of fault attacks, etc. and related documentation is publicly available.

Update process

Step 1: Submission Submission of an update of an existing mechanism or the addition of a new mechanisms is done internally at ECCG sub-group level, or by externals. External submissions are requested via the ENISA website: Ask questions, find answers - European Union Cybersecurity Certification. Use "request for ACM update" as the Subject and provide a short message to be further contacted by ENISA to deliver the update submission. The annex provides necessary elements to be included in an ACM update submission.

Step 2: Reception of an ACM update submission ENISA checks with the ECCG sub-group that the update submission is complete. Based on that, ENISA may:

- require more information to the submitter;
- invite the submitter to present in an ECCG Crypto subgroup session;
- provide comments to the submission.

Step 3: Analysis and update decision The ECCG sub-group analyses the update submission and additional information gathered according to step 2 and decides whether it is relevant or not for a future revision of ACM. If the update is not relevant, ENISA informs the submitter of its decision and justifies it. If the update is relevant, it is integrated in a next version of ACM.

Step 4: ACM update The ECCG sub-group can decide to update ACM when there are enough updates ready to be integrated or when an update is important (example: it has a strong impact on industry). Every two years, the ECCG sub-group will review ACM and check that all its elements are still applicable.

If an update is major (addition of a new mechanism for example), the document is submitted by ENISA to a public review of two months. After that review, the document is updated where necessary to consider the received comments, and the new version is adopted. If the update is minor (only editorial edits), the document is directly adopted by the sub-group.

Elements to be provided in a submission for an ACM update. The submission shall include following elements:

- **Contact of the submitter:** the ECCG sub-group may have additional questions and comments, so it is necessary to have a way to contact the submitter of the update suggestion.
- **Section/Quote:** identify as precisely as possible the section in ACM impacted by the change proposal, and quote the paragraph in question, if applicable.
- **Category:** indicate the nature of the proposed change: Editorial / Incorrect content / Question / New mechanism
- **Update presentation:** define the modification. If possible, submit a new text.
- **Rational for update and impact:** explain why this update is proposed and where applicable the impact for industry if it is included in ACM and if it is not.
- **References:** add any publication (mechanisms presentation, attacks, conference paper, etc.) that can support the update proposal.

For a new mechanism, the submission shall include in addition:

- associated standard(s): if the mechanism is not yet standardised, explain in which step of the standardisation process it is. Note that a non-standardised mechanism has few chances to be integrated in ACM;
- security proofs or security analysis by academics;
- side-channel analysis;
- fault injection attacks publications;
- practical use cases (including certified ones if any);
- any vulnerability published (bad/good implementations examples, etc.).

References

[ANSIX9.63] American National Standards Institute. *ANSI X9.63-2011 – Public Key Cryptography for the Financial Services Industry - Key Agreement and Key Transport Using Elliptic Curve Cryptography*. 2011.

[ANSSI-PG-083] Agence nationale de la sécurité des systèmes d'information. *Guide des mécanismes cryptographiques*. ANSSI-PG-083, version 3.00. 2026.

[BN00] M. Bellare and C. Namprempre. "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm". In: *ASIACRYPT*. Vol. 1976. Lecture Notes in Computer Science. Springer, 2000, pp. 531–545.

[ENISAAlgo] ENISA. *Algorithms, key size and parameters report*. 2014.

[ETSI] ETSI. *TS 103 744, Quantum-safe Hybrid Key Exchanges*. 2020.

[FIPS180-4] National Institute of Standards and Technology. *FIPS PUB 180-4: Secure Hash Standard (SHS)*. 2015.

[FIPS186-4] National Institute of Standards and Technology. *FIPS PUB 186-4: Digital Signature Standard (DSS)*. 2013.

[FIPS186-5] National Institute of Standards and Technology. *FIPS PUB 186-5: Digital Signature Standard (DSS)*. 2023.

[FIPS197] National Institute of Standards and Technology. *FIPS PUB 197: Advanced Encryption Standard (AES)*. 2001.

[FIPS202] National Institute of Standards and Technology. *FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. 2015.

[FIPS203] National Institute of Standards and Technology. *FIPS PUB 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard*. ~~draft~~. 2024.

[FIPS204] National Institute of Standards and Technology. *FIPS PUB 204: Module-Lattice-Based Digital Signature Standard*. ~~draft~~. 2024.

[FIPS205] National Institute of Standards and Technology. *FIPS PUB 205: Stateless Hash-Based Digital Signature Standard*. ~~draft~~. 2024.

[FIPS46-3] National Institute of Standards and Technology. *FIPS PUB 46-3: Data Encryption Standard (DES)*. 1999.

[FRODO-KEM] FrodoKEM Team. *FrodoKEM: Learning With Errors Key Encapsulation Preliminary Standardization Proposal*. 2023.

[HAC] A. Menezes, P. von Oorschot, and O. Vanstone. *Handbook of Applied Cryptography*. CRC Press. 1996.

[HybridTLS] D. Stebila, S. Fluhrer, and S. Gueron. *Hybrid key exchange in TLS 1.3, draft-ietf-tls-hybrid-design-10*. 2024.

[IS019772] ISO/IEC. *ISO/IEC 19772:2020 – Information technology – Security techniques – Authenticated encryption*. 2020.

[ISO10116] ISO/IEC. *ISO/IEC 10116:2017 – Information technology – Security techniques – Modes of operation for an n-bit block cipher*. 2017.

[ISO10118-3] ISO/IEC. *ISO/IEC 10118-3:2018 – Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*. 2018.

[ISO11770-3] ISO/IEC. *ISO/IEC 11770-3:2021 – Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*. 2021.

[ISO14888-3] ISO/IEC. *ISO/IEC 14888-3:2018 – Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms*. 2018.

[ISO18031] ISO/IEC. *ISO/IEC 18031:2011 – Information technology – Security techniques – Random bit generation*. 2011.

[ISO18033-2] ISO/IEC. *ISO/IEC 18033-2:2006 – Information technology – Security techniques – Encryption Algorithms – Part 2: Asymmetric ciphers*. 2006.

[ISO18033-3] ISO/IEC. *ISO/IEC 18033-3:2010 – Information technology – Security techniques – Encryption Algorithms – Part 3: Block ciphers*. 2010.

[ISO9796-2] ISO/IEC. *ISO/IEC 9796-2:2010 – Information technology – Security techniques – Digital signature schemes giving message recovery – Part 2: Integer factorization based mechanisms*. 2010.

[ISO9797-1] ISO/IEC. *ISO/IEC 9797-1:2011 – Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*. 2011.

[ISO9797-2] ISO/IEC. *ISO/IEC 9797-2:2021 – Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function*. 2021.

[JORF] ANSSI. "Avis relatif aux paramètres de courbes ellitpiques définis pas l'État français". In: *Journal Officiel* 0241 (Oct. 2011), p. 17533.

[L13] J. Lell. *Practical malleability attack against CBC-Encrypted LUKS partitions*. `http://www.jakoblell.com/blog/2013/12/22/practical-malleability-attack-against-cbc-encrypted-luks-partitions`. 2013.

[PKCS1] RSA Laboratories. *PKCS #1 v2.2: RSA Cryptography Standard*. 2012.

[RFC2104] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. 1997.

[RFC3526] T. Kivinen and M. Kojo. *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*. 2003.

[RFC5246] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. 2008.

[RFC5297] D. Harkins. *Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)*. 2008.

[RFC5639] M. Lochter and J. Merkle. *Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*. 2010.

[RFC5869] H. Krawczyk and E. Eronen. *HMAC-based Extract-and-Expand Key Derivation Function*. 2010.

[RFC7748] A. Langley, M. Hamburg, and S. Turner. *Elliptic Curves for Security*. 2016.

[RFC7919] D. Gillmor. *Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)*. 2016.

[RFC8017] K. Moriarty, B. Kaliski J. Jonsson, and A. Rusch. *Public-Key Cryptography Standard (PKCS) #1: RSA Cryptography Specifications Version 2.2*. 2016.

[RFC8018] K. Moriarty, B. Kaliski, and A. Rusch. *PKCS #5: Password-Based Cryptography Specification Version 2.1*. 2017.

[RFC8032] S. Josefsson and I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. 2016.

[RFC8446] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. 2018.

[RFC9106] A. Biryukov et al. *Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications*. 2021.

[RFC9370] CJ. Tjhai et al. *Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2)*. 2023.

[ROCA] Matus Nemec et al. "The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli". In: *24th ACM Conference on Computer and Communications Security (CCS'2017)*. ACM, 2017, pp. 1631–1648.

[S79] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (1979), pp. 612–613.

[SP800-185] National Institute of Standards and Technology. *SP800-185: SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash*. 2016.

[SP800-208] National Institute of Standards and Technology. *SP800-208: Recommandation for Stateful Hash-Based Signatures Schemes*. 2020.

[SP800-38A] National Institute of Standards and Technology. *SP800-38A: Recommendation for Block Cipher Modes of Operation*. 2001.

[SP800-38A-Addendum] National Institute of Standards and Technology. *SP800-38A-Addendum: Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. 2010.

[SP800-38B] National Institute of Standards and Technology. *SP800-38B: Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. 2005.

[SP800-38C] National Institute of Standards and Technology. *SP800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*. 2004.

[SP800-38D] National Institute of Standards and Technology. *SP800-38A: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. 2007.

[SP800-38E] National Institute of Standards and Technology. *SP800-38E: Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*. 2010.

[SP800-38F] National Institute of Standards and Technology. *SP800-38F: Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*. 2012.

[SP800-56A] National Institute of Standards and Technology. *SP800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*. Revision 3. 2018.

[SP800-56B] National Institute of Standards and Technology. *SP800-56B: Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography*. Revision 2. 2019.

[SP800-56C] National Institute of Standards and Technology. *SP800-56C: Recommendation for Key Derivation through Extraction-then-Expansion*. Revision 2. 2020.

[SP800-90A] National Institute of Standards and Technology. *SP800-90A Rev. 1: Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. 2015.

[TR-02102-1] Bundesamt fur Sicherheit in der Informationstechnik. *Technical Guideline TR-02102-1, Cryptographic Mechanisms: Recommendations and Key Lengths*. version 2026-01, 2026-01-23. 2026.

[TR-03111] Bundesamt fur Sicherheit in der Informationstechnik. *Technical Guideline TR-03111, Elliptic Curve Cryptography*. version 2.10, 01.06.2018. 2018.