SOG-IS Crypto Working Group

SOG-IS Crypto Evaluation Scheme
Agreed Cryptographic Mechanisms

Document purpose:	specify the requirements of the SOG-IS Crypto Evaluation Scheme related to the selection of cryptographic mechanisms. This document is primarily intended for developers and evaluators.

Version 1.0
May 2016

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 SOG-IS participants in the SOG-IS Crypto Evaluation Scheme.

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.

In the SOG-IS Crypto Evaluation Scheme, a cryptographic evaluation is always combined with the Common Criteria evaluation of a target of evaluation (TOE) whose security functionalities comprise some cryptographic mechanisms. A result of an evaluation performed under the SOG-IS Crypto Evaluation Scheme is that a user of the product can get assurance that, when following some guidance, the product only uses agreed cryptographic mechanisms to provide the security services evaluated under this scheme. These mechanisms are recognised to provide an acceptable security provided that they are implemented with sufficient precautions and their features adequatly address the TOE's security needs. Evaluation activities other than checking whether mechanisms are agreed, such as conformance testing, implementation evaluation, checking the overall consistency of the security architecture and key management of the TOE with its security goals, etc., are described in separate documents that together with this document constitute the SOG-IS Crypto Evaluation Scheme.

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 all 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.
- **legacy 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 gradually

phased out because they do no longer 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 legacy mechanisms. It is to be understood that after this validity period has expired, the mechanisms will no longer be considered as agreed mechanisms, and be accepted in later SOG-IS crypto evaluations. The default acceptability deadline for legacy mechanisms is set to (31 december) 2020.

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 SOG-IS Crypto WG will generally downgrade the status of a recommended mechanism to legacy, or speed up the expiration of some agreed legacy mechanisms. In exceptional cases, a decision may need to be made to remove an algorithm from both the recommended and the legacy 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 legacy status may be:

- 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.

An example of cryptographic breakthrough which will lead to a fast-track removal of some algorithms from the present document altogether might be:

- Implementation of a practical scalable universal quantum computer or of a quantum computer capable of solving any cryptographically relevant problem substantially faster than the fastest classical computers. Note that there are already algorithms designed for quantum computers breaking most usual asymmetric algorithms in polynomial time and speeding up generic attacks on symmetric algorithms (e.g. exhaustive search) by an exponential factor without allowing subexponential solving time.

While quantum computers do not represent an immediate threat to cryptography, this might happen in the future. The current document does not provide agreed quantum resistant mechanisms. Such mechanisms might be introduced in future versions since standardization of quantum resistant mechanisms is likely to take place within a few years.

For all the above reasons, developers of cryptographic systems with expected lifetimes longer than a few years should always take into account the possible need to migrate to newer algorithms, possibly including quantum-resistant algorithms.

In order for this document to stay relevant, it will be revised on 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 "legacy", the resulting mechanism is considered "legacy". 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 assymetric 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., 100 bits of security means that 2^{100} 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 tradeoff 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 2 keys (resp. 3 keys) is 112 bits (resp. 168 bits), strictly lower that the 128 bits (resp. 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 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. A future version of this document will also handle complex cryptographic protocols, such as TLS and IPsec. 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 *legacy* mechanisms for the considered type of mechanism. Each mechanism is either marked R (recommended) or L (legacy). If restrictions apply to the parameter values for which the mechanism is considered "R" or "L", 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.

1.5 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 in the SOG-IS Crypto Evaluation Scheme. 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-RGS] 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 $\{0,1\}^n$, selected by a k-bit parameter named the *key*: n and k represent respectively the block size and the key size, expressed in bits.

Agreed Block Ciphers.

Primitive	Parameters' sizes	R/L	Notes
AES [FIPS197, ISO18033-3]	k = 128 bits	R	
	k = 192 bits	R	
	k = 256 bits	R	
Triple-DES [FIPS46-3, ISO18033-3]	k = 168 bits	L	2-SmallBlocksize
	k = 112 bits	L	2-SmallBlocksize 3-TripleDES2key

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-TripleDES2key. Security of Triple-DES with two keys: $\min(2^{112}, 2^{120-t})$, with t such that 2^t is the number of input/output pairs of blocks available to the attacker. Thus for all uses of Triple-DES with two keys (encryption, message authentication, etc.) the amount of blocks processed with one particular key must be limited to 2^{20}.

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 a legacy scheme.

Agreed Hash Functions

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

2.4 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/L	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 7 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/L	Notes
CTR [SP800-38A, ISO10116]	R	6-StreamMode
OFB [SP800-38A, ISO10116]	R	6-StreamMode
CBC [SP800-38A, ISO10116]	R	7-Padding
CBC-CS (CiphertextStealing) [SP800-38A-Addendum]	R	
CFB [SP800-38A, ISO10116]	R	7-Padding

General Notes.

Note 4-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 5-AddIntegrity. The use of authenticated encryption schemes should be preferred over confidentiality only schemes.

Specific Notes.

Note 6-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 7-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 derministic encryption modes is tolerated.

Agreed Disk Encryption Schemes.

Scheme	R/L	Notes
XTS [SP800-38E]	R	9-UniqueTweak, 10-AddressTweak
CBC-ESSIV	L	11-UniqueSectorNumber, 12-AddressSectorNumber, 13-CBCMalleability

General Notes.

Note 8-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 9-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 10-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 9-UniqueTweak.

Note 11-UniqueSectorNumber. 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 12-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 11-UniqueSectorNumber.

Note 13-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 14-MACTruncation96. The truncation of a MAC generated by an agreed MAC mechanism to at least 96 bits is agreed.

A necessary condition, this may not be a sufficient condition for specific MAC schemes, e.g. GMAC.

Note 15-MACTruncation64. The truncation of a MAC generated by an agreed MAC mechanism to at least 64 bits is considered legacy under the condition that the maximal number of MAC verifications performed for a given key over its lifetime can be 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 23-GMAC-GCM-Bounds.

Agreed MAC Schemes based on a Block Cipher.

Scheme	R/L	Notes
CMAC [SP800-38B, ISO9797-1]	R	
CBC-MAC [ISO9797-1, Algorithm 1, Padding 2]	R	16-FixedInputLength

Specific Notes.

Note 16-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/L	Notes
HMAC [RFC2104, ISO9797-2]	$k \geq 125$	R	
	$k \geq 100$	L	
HMAC-SHA-1 [RFC2104, ISO9797-2, FIPS180-4]	$k \geq 100$	L	17-HMAC-SHA-1

Specific Notes.

Note 17-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 acceptable legacy 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.

Agreed Universal Hash Function Based MAC Schemes.

Scheme	R/L	Notes
GMAC [SP800-38D]	R	20-GMAC-GCMNonce 21-GMAC-GCMOptions 23-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 55-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 18-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	R/L
$\ell = 125$	R
$\ell = 96$	L

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/L	Notes
Encrypt-then-MAC [BN00]	R	19-DecryptionOrder
CCM [SP800-38C, IS019772]	R	19-DecryptionOrder
GCM [SP800-38D, IS019772]	R	19-DecryptionOrder 20-GMAC-GCMNonce, 21-GMAC-GCMOptions 22-GCMPlaintextLength 23-GMAC-GCM-Bounds
EAX [IS019772]	R	19-DecryptionOrder
MAC-then-Encrypt [BN00]	L	19-DecryptionOrder
Encrypt-and-MAC [BN00]	L	19-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, other 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 14-MACTruncation96 and 15-MACTruncation64 also apply to symmetric authenticated encryption schemes.

Specific Notes.

Note 19-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 20-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 21-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 one of the values 96, 104, 112, 120, and 128 bits.

Note 22-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 23-GMAC-GCM-Bounds. As the unforgeability bounds of the agreed GMAC and GCM options of 21-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/L	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/L	Notes
NIST SP800-56 ABC [SP800-56A, SP800-56B, SP800-56C]	R	
ANSI-X9.63-KDF [ANSIX9.63]	R	
PBKDF2 [RFC2898]	R	24-PBKDF2-PRF

Specific Notes.

Note 24-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

Agreed 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.

Scheme	R/L	Notes
PBKDF2 [RFC2898]	R	26-Salt, 25-NumberOfIterations

General Notes.

Note 25-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 26-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.

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. 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/L	Notes
RSA	$\log_2(n) \geq 3000$, $\log_2(e) > 16$	R	27-RSAKeyGen, 28-SmallD
	$\log_2(n) \geq 1900$, $\log_2(e) > 16$	L	

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

Note 27-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

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

This condition holds with overwhelming probability if p and q are randomly generated.

Note 28-SmallD. The size of d should be close to the size of n. Note that this is guaranteed for a small e. We should have at least $d > 2^{n/2}$, where n denotes the bitlength of the modulus.

Note that these conditions are also required by [FIPS186-4, Appendix B.3.1].

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 Primitive Sizes.

Primitive	Parameters' sizes	R/L	Notes
FF-DLOG	$\log_2(p) \geq 3000$, $\log_2(r) \geq 250$	R	
	$\log_2(p) \geq 1900$, $\log_2(r) \geq 200$	L	31-Precomputation

General Notes.

Note 29-CorrectSubgroup. It should be ensured that manipulated values have order divisible by r and dividing q. This ensures that they do not lie in subgroups of small size or outside the intended subgroup.

Note 30-PrimeOrder. It is advised to pick a subgroup of prime order, i.e. q should be prime. In this case, one has $r = q$ and checking 29-CorrectSubgroup boils down to checking that manipulated values have exact order r.

Specific Notes.

Note 31-Precomputation. Discrete logarithm algorithms involve a group related precomputation phase, whick 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. Another possibility is to generate fresh groups, e.g. following [FIPS186-4, Appendices A.1.1.2, A.2.2] with an agreed hash function.

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/L	Notes
Brainpool [RFC5639]	BrainpoolP256r1	R	
	BrainpoolP384r1	R	
	BrainpoolP512r1	R	
NIST [FIPS186-4, Appendix D.1.2]	NIST P-256	R	35-SpecialP
	NIST P-384	R	
	NIST P-521	R	
FR [JORF]	FRP256v1	R	

General Notes.

Note 32-PointOnCurve. Special precautions should be taken to ensure that manipulated points lie on the curve, i.e. they verify the curve equation.

Note 33-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 34-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 33-PointInSubgroup boil down to checking that the manipulated points have exact order r.

Specific Notes.

Note 35-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 (and not only because the arithmetic of the underlying finite field is faster).

4.4 Other Intractable Problems

The previous three mathematical problems are the most studied and the most widely used. Nonetheless there exist other mathematical problems leading to trapdoor or proof of knowledge constructions and for which no generic efficient solving method is known in general.

To cite a few:

- the discrete logarithm problem in (the group of rational points of the Jacobian of) hyperelliptic curves of genus two or three);
- lattices constructions such as the Learning With Errors construction (candidate for quantum resistance);
- 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);
- hash based cryptography (candidate for quantum resistance).

As aforementioned, these constructions, and/or aspects pertaining to their implementation, have received less attention than the integer factorization problem and the discrete logarithm problems in finite fields and elliptic curves and as a consequence they should only be used in a careful way. For the time-being, no asymmetic primitive beyond the three mathematical problems developed in this section is recognized agreed.

Since there is a perceived need to standardize quantum-resistant cryptography, the candidates for quantum resistant cryptography will most probably receive more attention in the coming years.

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 7 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/L	Notes
RSA	OAEP (PKCS#1v2.1) [RFC3447, PKCS1]	R	38-OAEP-PaddingAttack
RSA	PKCS#1v1.5 [RFC3447, PKCS1]	L	37-PaddingAttack

General Notes.

Note 36-RandomPadding. The asymmetric encryption schemes use a randomized padding that shall be generated by an agreed random bit generator, see Section 6.

Specific Notes.

Note 37-PaddingAttack. In case a padding oracle is available, the RSA-PCKS#1v1.5 scheme is vulnerable to efficient attacks.

Note 38-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.

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/L	Notes
RSA	PSS (PKCS#1v2.1) [RFC3447, PKCS1, ISO9796-2]	R	
FF-DLOG	KCDSA [ISO14888-3]	R	42-DSARandom
	Schnorr [ISO14888-3/am1]	R	
	DSA [FIPS186-4, ISO14888-3]	R	
EC-DLOG	EC-KCDSA [ISO14888-3]	R	42-DSARandom
	EC-DSA [FIPS186-4, ISO14888-3]	R	
	EC-GDSA [TR-03111]	R	
	EC-Schnorr [ISO14888-3/am1]	R	
RSA	PKCS#1v1.5 [RFC3447, PKCS1, ISO9796-2]	L	41-PKCSFormatCheck

General Notes.

Note 39-Hash. The scheme is agreed provided the underlying hash function is.

Note 40-DifficultProblem. The scheme is agreed provided the underlying mathematical problem uses agreed parameters.

Specific Notes.

Note 41-PKCSFormatCheck. Format checks should be carefully implemented to avoid attacks à la Bleichenbacher.

Note 42-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 6.3.

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 55-KeyUsage.

As for symmetric entity authentication schemes, the challenge should verify some properties, see 18-CollChallenge.

5.4 Key Establishment

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.

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 Schemes.

Primitive	Scheme	R/L	Notes
FF-DLOG	DH [SP800-56A, ISO11770-3]	R	44-DHAuth, 45-DHSubgroupAttacks
	DLIES-KEM [ISO18033-2]	R	
EC-DLOG	EC-DH [SP800-56A, ISO11770-3]	R	44-DHAuth, 45-DHSubgroupAttacks
	ECIES-KEM [ISO18033-2]	R	

General Notes.

Note 43-DifficultProblem. The scheme is agreed provided the underlying mathematical problem uses agreed parameters.

Specific Notes.

Note 44-DHAuth. DH is an 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 and identities. This authentication requires long-term secrets.

Note 45-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 29-CorrectSubgroup in the finite field case and 32-PointOnCurve, 33-PointInSubgroup in the elliptic curve case.

In the finite field case, selecting a group with prime order, as advocated in 30-PrimeOrder, makes the verifications easier.

6 Random Generator

6.1 Random Source

A 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 assurances 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 46-NoDirectRandomSource. No random source is considered agreed as is. For cryptographic applications, a deterministic random bit generator, eventually seeded by a random source, shall be used.

6.2 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 the random source, and from which pseudo-random bits can be extracted.

Agreed Deterministic Random Bit Generator.

Scheme	R/L	Notes
HMAC-DRBG [SP800-90A, ISO18031]	R	
Hash-DRBG [SP800-90A, ISO18031]	R	
CTR-DRBG [SP800-90A, ISO18031]	R	49-CTR-DRBG-BacktrackingResistance

General Notes.

Note 47-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 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 48-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 to produce ephemeral keys used in previous key exchanges will be able to break the PFS property if it is possible to practically compute past RBG output from the present state of the RBG. Therefore, it is recommended to use RBGs which do not allow such backwards computation.

Specific Notes.

Note 49-CTR-DRBG-BacktrackingResistance. For CTR-DRBG, the backtracking resistance property defined in 48-BacktrackingResistance is only given for the transition function between different invocations; it is therefore not advisable to derive a large number of random bits by a single invocation of CTR-DRBG.

6.3 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, constructions relying on such a random bit generator need to be implemented.

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/L	Notes
"Testing" technique [FIPS186-4, Appendix B.1.2]	R	
"Extra random" technique [FIPS186-4, Appendix B.1.1]	R	

General note.

Note 50-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 generator. The generation of asymmetric parameters and RSA keys requires to generate random prime numbers. The generation of random primes follows the following strategy: initially an integer of the appropriate size is drawn at random. Then, it is tested for primality. As long as the primality test fails, 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.

Note 51-Probableprime. In case the prime generation method does not prove the primality of its output, the probability that this output is composite should be lower that 2^{-125}.

7 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 52-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 52-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.

7.1 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	53-KeyEstablishment,54-KeyGenerationSeed
Agreed key derivation function	54-KeyGenerationSeed

Note 53-KeyEstablishment. A key established through key establishment should not be used directly but first postprocessed by an agreed key derivation function.

Note 54-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 large enough (at least 125 bits for recommended mechanisms, or at least 100 for legacy mechanisms).

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 55-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.

7.2 Key Storage and Transport

Note 56-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.

7.3 Key Use

Note 57-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 58-Diffusion. The diffusion of secret and private keys shall be limited to the trusted environment that makes an effective use of the key.

7.4 Key Destruction

Note 59-KeyDestruction. After the use period of a key has expired, it 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 exchanges 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 remanence issues.

8 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 herself 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 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 succed 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 6-digit PIN. 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 between every unsuccessful authentication attempt. 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: corroboration 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.

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-RGS] Agence nationale de la sécurité des systèmes d'information. *Annexes B du référentiel général de sécurité*. version 2.03. 2014.

[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.

[FIPS180-4] National Institute of Standards and Technology. *FIPS PUB 180-4: Secure Hash Standard (SHS)*. 2012.

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

[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.

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

[IS019772] ISO/IEC. *ISO/IEC 19772:2009 – Information technology – Security techniques – Authenticated encryption*. 2009.

[ISO10116] ISO/IEC. *ISO/IEC 10116:2006 – Information technology – Security techniques – Modes of operation for an n-bit block cipher*. 2006.

[ISO10118-3] ISO/IEC. *ISO/IEC 10118-3:2004 – Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions*. 2004.

[ISO11770-3] ISO/IEC. *ISO/IEC 11770-3:2008 – Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*. 2008.

[ISO14888-3] ISO/IEC. *ISO/IEC 14888-3:2006 – Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms*. 2006.

[ISO14888-3/am1] ISO/IEC. *ISO/IEC 14888-3:2006/am1:2010 – Information technology – Security techniques – Elliptic Curve Russian Digital Signature Algorithm, Schnorr Digital Signature Algorithm, Elliptic Curve Schnorr Digital Signature Algorithm, and Elliptic Curve Full Schnorr Digital Signature Algorithm*. 2010.

[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 9797-2:2011 – Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function*. 2011.

[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.

[RFC2898] B. Kaliski. *PKCS #5: Password-Based Cryptography Specification Version 2.0*. 2000.

[RFC3447] J. Jonsson and B. Kaliski. *Public-Key Cryptography Standard (PKCS) #1: RSA Cryptography Specifications Version 2.1*. 2003.

[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.

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

[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*. 2013.

[SP800-56B] National Institute of Standards and Technology. *SP800-56B: Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography*. 2009.

[SP800-56C] National Institute of Standards and Technology. *SP800-56C: Recommendation for Key Derivation through Extraction-then-Expansion*. 2011.

[SP800-90A] National Institute of Standards and Technology. *SP800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. 2012.

[TR-02102-1] Bundesamt fur Sicherheit in der Informationstechnik. *Technische Richtlinie TR-02102-1, Kryptographische Verfahren: Empfehlungen und Schlussellangen*. version 2015-01, 10.02.2015.

[TR-03111] Bundesamt fur Sicherheit in der Informationstechnik. *Technical Guideline TR-03111, Elliptic Curve Cryptography*. version 2.0, 28.06.2012.