Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

PRIME MINISTER

General Secretariat for
National Defence
Central Directorate for
Information Systems Security

Paris, 2007 september 14

No. 1904/SGDN/DCSSI/SDS/LCR

Cryptographic mechanisms

Rules and recommendations about the choice and parameters sizes of cryptographic mechanisms with *standard* robustness level

Version 1.10

Version	Modifications
Version 1.10	First translated version. Main modifications made: - creation of the “key management” reference documentation taken into account; - indication of the document describing supplies necessary to evaluate cryptographic mechanisms; - changes in the state-of-the-art taken into account in the following subjects: • stream cipher algorithms; • block cipher algorithms (Recom$_{E}$AlgoBloc-1); • encryption mode of operation; • asymmetric mathematical problems; • generation of random numbers. - comment added about the status of SHA-1.
Version 1.02, October 25 2004	First applicable version. Available only in French.

DCSSI, Laboratory of Cryptography

SGDN/DCSSI/SDS/Crypto

51 boulevard de la Tour-Maubourg, 75700 Paris 07 SP

Crypto.DCSSI@sgdn.pm.gouv.fr

A.Table of contents

B. Introduction

Modern cryptography provides information system designers with tools for assuring, or contributing to assuring security functions such as confidentiality, integrity, authenticity and non-repudiation. These tools are often described as being algorithms, primitives or **cryptographic mechanisms**.

Subsequent to major developments that took place in the 1980s and 1990s, the science of cryptography (although still young) has apparently reached a sufficient degree of maturity so that general rules about the choice and correct use of mechanisms can be identified. This document aims at explaining these rules as well as some recommendations.

B.1. Document purpose

This document describes rules and recommendations about the first level of cryptographic robustness. These levels are defined below under B.3.

Important note: this document is publicly available in two versions:

- **this version of the document is not classified; it only deals with the first robustness level, called the "standard level". This document is meant to be widely distributed, particularly as an electronic document.**
- a second version of the document is also not classified, but is not distributed as an electronic document; it deals with the first two robustness levels, namely "standard" and "reinforced" levels. The information contained in it is derived from public know-how and state-of-the-art cryptography. This version of the document is currently available only in French.

Rules define principles that must a priori be followed by any mechanism aiming at a given robustness level. Compliance with these rules is a condition that is usually necessary but is not sufficient for recognition of the robustness level targeted by the mechanism. On the other hand, following all the rules, which are inherently very generic ones, does not guarantee the robustness of the cryptographic mechanism; the only way to assess this robustness is to carry out a specific analysis.

We define not only rules, but also **recommendations.** The purpose of these recommendations is to guide the user in choosing some primitives and to suggest some principles for the choice of parameters sizes to provide considerable gain in security, often at a reasonable cost. Obviously, there is more freedom associated with the application of recommendations, depending on other requirements such as performance constraints.

We emphasize that the rules and recommendations contained in this document are not intended to be a dogma for designers of products based on cryptographic mechanisms. Our purpose is to contribute to continuous improvements of the quality of security products. In this respect, following rules set out in this document should be considered as a healthy approach to guard against a variety of design errors and other possible weaknesses not detected during the evaluation of cryptographic mechanisms.

For transparency reasons, we tried to justify each rule and recommendation contained in this document. The purpose is to convince the reader that choices have not been made arbitrarily, but rather taking account of the current state-of-the-art in cryptography and practical constraints identified during its implementation, as rigorously as possible.

The definition of rules and recommendations is based on existing state-of-the-art in cryptography. Considering this and some classical assumptions, such as Moore's law about the advance in available computing power, enables to define rules and recommendations sufficiently objectively and scientifically to establish a framework that would be acceptable to any specialist in information system security. However, it is quite clear that such an analysis cannot take account of "catastrophic" events such as an operational cryptanalysis of AES, or the discovery of an effective method for factorizing large numbers.

Furthermore, it is difficult to determine what resistance levels will be necessary to guarantee information security in 10 to 20 years. This, however, is necessary for many applications such as the protection of confidentiality of some information, or electronic signature applications for which long term validity is often necessary. Furthermore, when defining a product, the envisaged product life determines whether the required visibility must be short, middle or long term. Obviously, some problems could be solved by technical means (regular superencryption of information to be protected in the long term, time stamping and regular signature of notarial documents, etc.); this approach is sometimes essential, but it cannot be applied generally due to the constraints it implies. Consequently, we have tried to develop an analysis that will be valid for more than fifteen years, although this involves risks. This can easily be seen by comparing the current state-of-the-art with the state-of-the-art a few decades ago; no currently used mechanism is more than thirty year old, and the oldest used mechanism is undoubtedly the DES, which was standardized in 1977.

B.2. Limits to the scope of the document

This document deals with rules and recommendations about the choice and parameters sizes of cryptographic mechanisms. It should be read in conjunction with the document entitled "Management of cryptographic keys – Rules and recommendations about the management of keys used in cryptographic mechanisms", which deals more specifically with the aspects related to creation, distribution and manipulation of keys. Therefore, these aspects are not dealt with in this document. This document also excludes specifically:

- recommendations about precise cryptographic mechanisms known to achieve the different cryptographic robustness levels defined in this document, although some very classical standard level primitives are mentioned,
- aspects related to the implementation of mechanisms, and particularly the choice of the hardware and the security of the installation against side channel attacks (timing attack, SPA, DPA, HODPA, etc.) or by fault injection (DFA),
- aspects related to possible confidentiality of mechanisms and to the need to diversify these mechanisms depending on usage contexts,
- methods for evaluating cryptographic mechanisms, which are based above all on a precise knowledge of the state-of-the-art in cryptography,
- methods of evaluating threats and designing cryptographic products in order to choose cryptographic mechanisms for performing identified security functions and necessary cryptographic robustness levels,
- relations between the robustness level of a cryptographic mechanism and the robustness level of a product as defined in qualification or evaluation procedures according to a standard method such as Common Criteria,
- supplies necessary to evaluate cryptographic mechanisms, described in a separate document[1] entitled "Supplies necessary for analysis of cryptographic mechanisms – version 1.2" No. 2336/SGDN/DCSSI/SDS, November 6 2006,
- non-cryptographic mechanisms, which nonetheless perform security functions such as the use of passwords or biometry, etc. These mechanisms are not included in the scope of this document because they cannot be analyzed using usual cryptographic methods. However, this does not question their possible advantages in some applications.

B.3. Cryptographic robustness levels

This document defines **two levels of cryptographic robustness**, but later on the document only describes the rules and recommendations for the first level. These different levels naturally appear when dealing with the resistance of cryptographic mechanisms. These two levels will be referred to using the same terms as in the qualification, namely "standard" and "reinforced" levels.

Note that rules and recommendations defined in this document do not take account of the conditions for use of cryptographic mechanisms. Such an approach may be surprising because it differs from the approach used for evaluating the robustness level of products and systems. Its main advantage is that it takes account of most attack scenarios, including scenarios that might have been forgotten or neglected by product designers. This approach is also justified by the existence of mechanisms that are guaranteed to be secure, including in a very restrictive environment, without necessarily requiring many more resources than second-rate primitives.

[1] This document is currently available only in French.

Relation between the cryptographic robustness level and the product qualification level

Robustness levels are applicable to cryptographic mechanisms and not to the products that use them. The purpose of this document is to help assessing the intrinsic robustness of mechanisms, in this initial phase, without taking into account particular conditions of use of mechanisms in products. In particular, this method makes it possible to reuse results of cryptographic assessments from one product to another. Experience also shows that this method of assessment provides a means of detecting attack paths that would be difficult to imagine if the initial approach only considered apparently realistic scenarios.

The purpose of a cryptographic mechanism is to provide a security function in a given environment and in combination with other mechanisms. Thus for example, it is a priori be conceivable to use a standard-level mechanism within a product requiring a reinforced-level qualification, when it is really justified.

Based on the same approach, it must obviously be realized that the composition of cryptographic mechanisms with a given robustness level does not absolutely guarantee that a global equivalent robustness level will be obtained.

The choice of cryptographic primitives with a robustness level equal to at least the global level targeted for the product that uses them, is undoubtedly highly desirable in the general case, but it is therefore neither necessary nor automatically sufficient to guarantee product security. Designers and evaluators of cryptographic products must keep this fact in mind. **However, as a general rule, it is a good idea that the robustness level of the mechanisms used should be at least equal to the robustness level of the associated product.**

Relation between the cryptographic robustness level and the classification level of the information processed

The fact that two mechanism robustness levels are defined should not suggest that they should automatically be associated with “restricted” and “confidential” information classification levels, although there is obviously a natural link. In accordance with what is said in the previous paragraph, the choice of the robustness level required to protect information at a given classification level can only be made taking account of the threat, environment, installation constraints, etc.

Definitions of robustness levels

We will now define the two cryptographic robustness levels.

Standard level (definition)

The so-called “**standard**” cryptographic robustness level is obtained using mechanisms for which it has been impossible to demonstrate an attack in a realistic security model, even by a highly skilled attacker with very powerful computational capacities, typically of the order of magnitude of that available to major nations.

The attacker’s point of view is given priority at this level; therefore a standard robustness level does not necessarily imply that the mechanism is fully satisfactory in cryptographic terms. This is the least restrictive level, to guarantee cryptographic security. The main risk associated with such an approach is to create a vulnerability to realistic attacks not detected during the evaluation.

Reinforced level (definition)

The so-called “**reinforced**” cryptographic robustness level is obtained using standard level mechanisms, used together with rules for good cryptographic practice derived from modern state-of-the-art in academic cryptography.

Obviously, it is very difficult to exhaustively list or very precisely define these “rules of good practice”, which form part of the know–how of experts in the field. However, modern cryptography has reached a sufficient degree of maturity so that in practice it is easy to reach an agreement about these rules.

Lack of a robustness level lower than the “standard” level

The first defined robustness level, namely the standard level, can reach a security level that can be considered to be very high, or even too high, in practice. In particular, we do not define a robustness level that can be described as “medium”, in the sense that mechanisms with such a level could potentially be attacked in operation without requiring major computation means and/or expertise capacities. This choice is justified by the fact that modern cryptography offers mechanisms for providing at least the standard level in all fields, at a cost in terms of computation time, required memory and difficulty of installation, comparable with the corresponding cost for any mechanisms that might be described as “medium” level. Cryptography is special in that it is often fairly easy to achieve a high robustness level for a limited extra cost. Furthermore, many recommendations contained in this document are related to the observation that a considerable security gain can often be obtained at minimum cost. Note that obviously, this reasoning cannot be applied to all aspects of information system security, and is apparently specific to cryptography. Therefore, we have decided not to define a “medium” level, the purpose of which would simply be to label mechanisms for which an evaluation would have shown that that they do not achieve the standard robustness level.

In conclusion, this document was written based on the intention to use cryptographic mechanisms ensuring high security. This approach is very classical in cryptography. Therefore, the threats considered are threats caused by realistic but very powerful attackers. A direct consequence is that no attempt is made to define sizes of keys or parameters that would simply provide protection from the least powerful attackers. Such estimates aimed at building a cryptographic system in which the target is slightly beyond the limit of the capacities of "threatening" attackers could possibly be justified in a "multi-origin" risk analysis. However, such an approach is much too specific to be developed in a generic document.

B.4. Document organization

This document is organized as follows:

- all rules and recommendations are grouped in chapter C, starting from page 12 ; they are identified using the following coding system: the first letters (Rule or Recom) identify whether a rule or a recommendation is being given, the next index (S) indicates the standard robustness level, followed by the scope and finally a figure to distinguish between rules in the same category. For example **Rule$_{\mathbf{s}}$Fact-3** denotes standard level rule number 3 related to the factorization problem;
- bibliographic references appear in page 31 ;
- the appendix contains a non-mathematical reminder of the main cryptographic concepts necessary to understand this document, starting on page 33 ;
- information derived from academic publications about the size of cryptographic mechanisms is given in another appendix starting on page 52.

Deliberately, this document does not contain any summary table showing minimum parameter sizes required for the standard robustness level. Conciseness has been preferred in the expression of rules and recommendations ; attempting to summarize them as a simple numeric value would be a serious source of error and confusion.

B.5. Document update and classification

The main purpose of this document is to fix numeric limits, for example in terms of key sizes, and it should be updated regularly. An annual revision would appear to be both realistic and sufficient. The DCSSI[2] cryptography laboratory will collect comments and distribute revisions.

[2] To contact us, please use the mailing or email address (not secure) on page 3 of this document.

C. Rules and recommendations

The rules and recommendations contained in this document are organized in a manner very similar to the cryptographic reminders given in appendix F, starting on page 33. They are aimed at a reader familiar with these concepts, which are thus not systematically repeated.

C.1. Symmetric cryptography

C.1.1. Symmetric encryption

C.1.1.1. Symmetric key size

This section defines the expected properties of keys used by symmetric mechanisms. Key size means the number of effective bits in the key, i.e. the number of genuinely variable bits[3]. For example, the DES uses 64-bit keys but only 56 of these key bits can be chosen at random, the remaining 8 bits being used as parity check. This is why it is assumed that DES keys have a size of 56 bits.

The minimum sizes defined below are only valid based on the assumption that the best practical attack to defeat the used symmetric mechanism consists in making an exhaustive search on the key space. Since this attack is generic, respecting the rules defined below is a necessary condition but it cannot be considered as being sufficient. In particular, carrying a cryptographic analysis is essential.

Standard level

Rule$_S$SymKey-1. The minimum size of symmetric keys to be used until 2010 is 80 bits.

Rule$_S$SymKey-2. The minimum size of symmetric keys to be used beyond 2010 is 100 bits.

Recom$_S$SymKey-1. The minimum recommended size of symmetric keys is 128 bits.

Justification:

- There is a great deal of controversy on how to estimate the computation capacity that a motivated organization can draw upon. However, many indications (see F.1, G1.1 and G.2.1) suggest that it will be impossible to attack 80-bit keys before 2010.
  - The use of keys containing fewer than 80 bits would appear to be risky. 56-bit keys are clearly insufficient and it is generally accepted at the moment that 64-bit keys can be attacked, although not everyone can make such a computation. However, such attacks have been made, specifically in the public environment (see G.1.1).

[3] Formally, the key size is defined as a function of the possible key space and the probability of the choice of each of these keys using the entropy concept. In practice, such a complex approach is usually pointless, the intuitive concept of an "effective key size" being obvious.

- It still appears difficult to imagine that an exhaustive search could be made on 100-bit keys before several decades. However, the use of keys containing less than 128 bits should tend to disappear with the use of modern algorithms such as the AES.

Comments:

- The impact of the use of keys containing not less than 128 bits in terms of performances is often low, as demonstrated in the AES example.
- The use of 128-bit keys can assure that generic exhaustive search attacks will not work, even in the fairly long term. Obviously this does not mean that all mechanisms using such keys are cryptographically reliable.
- The use of 112-bit keys, as in the case of the triple DES, does not cause any practical security problem against exhaustive search attacks. However, it might be recommended that the triple DES should not be used for other reasons, particularly related to the block size that is not sufficient (64 bits) to provide practical security with some classical modes of operation.

C.1.1.2. Block cipher

The simplest two characteristics of a block cipher mechanism are the effective size of the key and the size of the processed blocks (see F.1.1.1). The previous section gives rules and recommendations about the effective key size.

C.1.1.2.1. Block size

Standard level

Rule$_S$BlockSize-1. The minimum block size in block cipher mechanisms is 64 bits.

Recom$_S$BlockSize-1. The recommended block size in block cipher mechanisms is 128 bits.

Justification:

- The use of too small blocks makes elementary attacks such as the composition of dictionaries more efficient in practice than the search for the secret key. It is commonly accepted that the size of a block must be at least 64 bits.
- Block cipher mechanisms are used through modes of operation to encrypt arbitrary messages or to calculate message authentication codes. The block size is then used in the estimate of the security of these mechanisms. The main threat is the frequent discovery of so-called "*square root*" attacks, or "*birthday paradox*" attacks (see F.1.3, section 57); this means that some attacks become operational as soon as $2^{n/2}$ message blocks have been processed, where n denotes the block size in bits. Therefore, the security limit for 64-bit blocks is only a few billion blocks, which can be quickly reached for some applications. In practice; a simple method of guarding against such attacks is to use 128-bit blocks.

C.1.1.2.2. Choice of the algorithm

The choice of a block cipher algorithm is based on the use of rules and recommendations related to the key size and to the block size. Apart from the simple consideration of these two dimensions, it is obviously necessary to consider the intrinsic security provided by the mechanism faced with more advanced attacks than a simple exhaustive search on the key (linear cryptanalysis, differential cryptanalysis, etc.).

Consider an attack on a block cipher algorithm. Such an attack has a goal and requires some means. Its goal may be to find the key, however it may be more modest and simply aim to distinguish the block cipher from a random permutation. As for the means of the attack, they may for example consist in allowing the attacker to observe encrypted data, the corresponding plain data being known or unknown, or allowing him to have messages of his choice encrypted, or even to have chosen data decrypted.

In order to simplify the matter, it is usually assumed that an attack is qualified by:

- the number N_{op} of operations necessary for the attack, one operation being equivalent to one block encryption,
- the number N_{block} of blocks to be encrypted or decrypted in order to make the attack,
- the amount N_{mem} of memory necessary, for example to store preliminary computations.

Standard level

Rule$_S$BlockAlgo-1. For an encryption algorithm that does not have to be used after 2010, there must be no known attack requiring less than $N_{op}=2^{80}$ operations

Rule$_S$BlockAlgo-2. For an encryption algorithm that is to be used after 2010, there must be no known attack requiring less than $N_{op}=2^{100}$ operations

Recom$_S$BlockAlgo-1. It is recommended that block cipher algorithms widely tested in the academic environment should be used.

Important comment:

- The rules do not mention the number of blocks N_{block} to be encrypted or decrypted in order to make the attack, nor the required memory amount N_{mem}. This is essentially dependent on the desire to not excessively complexify the statement of these rules. A judgment should be made for each case about whether one of these two parameters is sufficiently important to justify that a block cipher mechanism achieves the standard level even if it does not comply with the **Rule$_S$BlockAlgo-1** and/or **Rule$_S$BlockAlgo-2** rules.

Justification:

- Rules attempt to define attacks that are usually described as being practical, operational or realistic, although these terms often have different meanings depending on who uses them.
- The practical aspects of attacks are given preference for the standard robustness level. However, it is obvious that the existence of more “theoretical” attacks on a given mechanism demonstrates inherent weaknesses, even if they do not directly lead to operational attacks.
- For the standard level, the use of algorithms that have been widely studied by the academic community provides a very important guarantee of quality.

Standard level mechanism(s):

- The AES as specified in the FIPS-197 is a standard-level block cipher mechanism.

Comments:

- The triple DES, i.e., the use of the DES with two keys K_1 and K_2, encrypting with K1, decrypting with K2 and then re-encrypting with K_1, is a block cipher algorithm using 112-bit keys and 64-bit blocks. As a block cipher mechanism, the triple DES meets the standard level. However, this mechanism does not follow the recommendations given above in terms of key size and block size. Furthermore, one should be extremely cautious when using this mechanism with an encryption or integrity mode of operation, or in key transport protocols, particularly due to the small block size.

C.1.1.2.3. Encryption mode of operation

The encryption mode of operation assures confidentiality of arbitrary sized messages starting from a block cipher primitive. As explained in F.1.1.1, a simple block cipher mechanism cannot perform such a function, particularly due to its fundamentally deterministic nature and the imposed size of processed data blocks.

Standard level

The choice of an encryption mode of operation depends closely on the nature of the data being processed and the security model envisaged for this mechanism. At standard level, rules and recommendations are relatively generic.

Rule$_S$EncMode-1. There must not be any attack with complexity less than $2^{n/2}$, where *n* is the block size in bits, within the security model associated with use of the encryption mode.

Recom$_S$EncMode-1. The use of a randomized encryption method is recommended.

Recom$_S$EncMode-2. An encryption method will preferably be used jointly with the use of an integrity mechanism. Such a mechanism can be independent of the encryption method.

Justification:

- Many methods such as the CBC (see F.1.1.1, page 40) are only reliable if no more than the order of $2^{n/2}$ plaintext message blocks are processed, where *n* denotes the block size in bits. This limit can be quickly reached for an encryption mechanism using 64-bit blocks. An even lower bound would seriously compromise the security of the encryption mechanism.
- The practical aspects of attacks are given preference for the standard robustness. However, it is obvious that the existence of more “theoretical” attacks on a mechanism demonstrates inherent weaknesses, even if they do not directly lead to operational attacks.
- In order to assure confidentiality of the information, an encryption method must not be deterministic so that encryption of a particular message will not always give the same ciphertext. The use of “initial values” and an appropriate mode of operation (see F.1.1.1) can solve this problem.
- It is important to take account of the capacity of an attacker to observe encrypted messages, and also to obtain the corresponding plaintexts, to encrypt or decrypt messages of his choice, etc. In this case the security model is fundamental: restricting an analysis to the scenario in which the attacker only sees ciphertexts is a serious error that can have practical consequences on the security of the mechanism.
- The need for confidentiality is often associated with a need for integrity, even if integrity is not as obvious at first sight. It is very important to be aware of the fact that an encryption mechanism can provide a very high level of protection in terms of

confidentiality without providing any integrity at all! In particular, none of the classical methods (ECB, CBC, OFB, CFB, CTR) provides the slightest protection in terms of integrity.

Standard level mechanism(s):

- The CBC encryption mode of operation using a standard encryption primitive such as the AES, with initial values chosen at random for each message and transmitted in plain text is a symmetric standard level encryption mechanism. This mechanism is described on page 40.

C.1.1.3. Stream cipher

Stream cipher algorithms[4] form another large family of symmetric encryption mechanisms (see F.1.1.2).

C.1.1.3.1. Choice of the algorithm

Before defining rules related to the choice of stream cipher algorithms, we should mention that in general these algorithms do not provide any integrity at all for the transmitted messages (see above comment for block cipher methods). However, for a stream cipher algorithm without feedback, decryption of an encrypted message not generated by the encryption algorithm does not provide any additional information that could facilitate an attack. The security model usually selected for evaluating such mechanisms is that the adversary knows the output stream from the algorithm.

Furthermore, the following rules and recommendations regarding stream cipher are based on the observation that most academic proposals at the moment, even recent proposals, have been cryptanalyzed. However, these rules might be revised if major theoretical progress is made in the stream cipher field.

Standard level

Rule$_S$StreamC-1. For a stream cipher algorithm that does not have to be used after 2010, there must be no known attack requiring less than 2^{80} operations.

Rule$_S$StreamC-2. For a stream cipher algorithm that does have to be used after 2010, there must be no known attack requiring less than 2^{100} operations.

Recom$_S$StreamC-1. It is recommended that the stream cipher algorithms used should have been widely tested in the academic environment.

Recom$_S$StreamC-2. It is recommended that block algorithms rather than stream cipher algorithms shall be used. Notably, if stream cipher properties are required, a standard level block cipher method simulating a stream cipher can be used.

[4] The so-called "one-time pad" algorithm, which is simply a bitwise addition of the message to be encrypted and a key of the same size, is a special case in the classification of encryption algorithms. In particular, it cannot be considered as being a stream cipher even if stream ciphers are often derived from it. This is the only algorithm to have perfect security. However its usage constraints are such that it is impossible to use it in practice, except in very special cases. Remember that this mechanism in particular requires a one-time key with the same length as the message to be protected, and that it is absolutely impossible to reuse this key. In general, the use of the one-time pad algorithm simply changes the problem of encryption to a problem of key agreement.

Important comment:

- Rules do not mention the amount of data to be encrypted or decrypted to make the attack, nor the required amount of memory. This is mainly due to the need to avoid excessive complexity in the statement of these rules. A judgment should be made for each individual case about whether or not one of these two parameters is sufficiently important to justify that a stream cipher mechanism provides the standard level, even if it does not comply with the **Rule$_S$StreamC-1** and / or **Rule$_S$StreamC-2** rules.

Justification:

- The practical aspect of attacks is given preference for the standard robustness. However, it is obvious that the existence of more "theoretical" attacks demonstrates inherent weaknesses with a mechanism, even if they do not lead directly to operational attacks.
- For the standard level, the use of algorithms that have been widely studied by the academic community provides a very important guarantee of quality.
- Stream cipher algorithms are sometimes preferred to block cipher algorithms for their efficiency. However, experience shows that it is very difficult to design stream cipher algorithms. Many proposals have been cryptanalyzed and know-how does not seem to be sufficiently mature compared with know-how in the block cipher field. Consequently, we recommend the use of block cipher combined with satisfactory methods, in preference to stream cipher algorithms.
- When the stream cipher properties are specifically required, it is recommended that a block cipher method with a comparable behavior should be used, for example such as the classical OFB, CFB or CTR methods, if the degradation in performance is acceptable. Know-how acquired for block cipher methods can thus be used, while benefiting from a mechanism comparable to a stream cipher.

C.1.2. Authentication and integrity of messages

Rules on message authentication and integrity methods are very dependent on the chosen mechanism. However, some very general rules can be given.

Standard level

Rule$_S$IntegSym-1. The most classical symmetric integrity methods are based on block cipher or hashing mechanisms. Such primitives should have the standard robustness level.

Comments:

- **As a result of confusion with encryption methods, the use of "initial values" is often observed for integrity mechanisms such as the CBC-MAC (see page 44); this may create serious security weaknesses.**
- It is important to take account of the capacity of an attacker to observe integrity elements but also to obtain such elements, for example for messages of his choice.
- Many methods, such as CBC-MAC are only reliable if no more than the order of $2^{h/2}$ message blocks are processed, where h is the block size in bits. This limit can be reached quickly for an encryption mechanism using blocks with $h = 64$ bits.

- **The use of large keys does not necessarily guarantee a corresponding security level. Thus, most variants of the CBC-MAC created to obtain security comparable to the triple DES have been cryptanalyzed, in the sense that their security is more comparable to the DES than to the triple DES** (this is the case in the example on page 44 if the DES is used as the block cipher algorithm, even if the total key size is 112 bits)**.**
- An integrity mechanism is often used in addition to a confidentiality mechanism. A combination of two cryptographic mechanisms is never simple and must be made carefully. For example, it is easy to combine a very good encryption with a very good integrity algorithm and create a mechanism that no longer performs the confidentiality service.

<u>Standard level mechanism:</u>

- The "retail" CBC-MAC integrity method using AES as the block cipher mechanism and two distinct keys (one for the CBC chain and the other for so-called "retail" encryption) is standard level (obviously provided that an initial value is not used). This mechanism is described on page 44.

<u>Mechanism not providing the standard level:</u>

- The "retail" CBC-MAC integrity method recommended above does not reach the standard level if it is used with the DES as the block cipher mechanism, even if it uses two distinct keys. Although 112 key bits are used, observation of 2^{32} valid MACs provides a means of finding these 112 key bits by carrying out "only" of the order of 2^{56} DES computations.

C.1.3. Authentication of entities

Mechanisms using passwords in any form whatsoever (pass-phrase, personal identification code, etc.) and mechanisms based on biometric processes are not cryptographic, and consequently are not dealt with in this document. Obviously, this does not mean that they are not useful for making an information system secure.

However, it is worth mentioning some elementary comments related to the use of passwords:

- if it is required to derive secret keys from passwords, the passwords must be sufficiently long and must not be "guessable", so as to achieve a security level compatible with the rules about key sizes. For example, passwords with 8 alphanumeric characters (numbers and upper or lower case letters) cannot be used to generate keys containing more than 47 bits, and even then based on the very optimistic assumption that these passwords are chosen at random.
- When studying the security of an authentication mechanism, a distinction should be made between computations that can be done "off-line", i.e. computations that can be carried out without access to a resource such as a server, and operations that must be done "on-line". Thus, a password authentication protocol must not enable effective "off-line" verification tests. The number of resources that could be attacked also has to be considered.

- Finally, note that “birthday paradox” attacks can also be used, for example if a list of system access password hashes is available.

Furthermore, as mentioned in F.1.3, interactive entity authentication mechanisms are generally based on random number generation mechanisms and encryption mechanisms; therefore, the rules mentioned in sections C.1.1 and C.3.2 are directly applicable. Obviously, the global robustness level of the mechanism must be evaluated carefully, even if the primitives used have a compatible level.

C.2. Asymmetric cryptography

All asymmetric cryptography mechanisms are based on standard algorithmic problems, usually derived from the number theory (see F.2, page 46). Consequently, the use of problems that are genuinely difficult for an attacker is essential in terms of security.

C.2.1. Asymmetric mathematical problems

C.2.1.1. Factorization

The RSA factorization problem consists in finding the decomposition into prime numbers of an integer obtained secretly by the multiplication of two comparably sized prime numbers. Such a composite number is classically called a “modulus”.

Furthermore, the factorization problem is usually encountered using the RSA cryptosystem. Two other data items called the “public exponent” and the “secret exponent” are then used.

Standard level

Rule$_S$Fact-1. The minimum modulus size for use not beyond the year 2010 is 1536 bits.

Rule$_S$Fact-2. The minimum modulus size for use beyond the year 2010 is 2048 bits.

Rule$_S$Fact-3. The minimum modulus size for use beyond 2020 is 4096 bits.

Rule$_S$Fact-4. Secret exponents must be the same size as the modulus.

Rule$_S$Fact-5. For encryption applications, public exponents shall be greater than $2^{16} = 65536$.

Recom$_S$Fact-1. It is recommended that moduli with at least 2048 bits should be used, even for use not beyond 2010.

Recom$_S$Fact-2. It is recommended that public exponents larger than $2^{16} = 65536$ should be used for all applications.

Recom$_S$Fact-3. It is recommended that the two prime numbers p and q forming the modulus should be the same size and should be generated at random.

Justification:

- The size of RSA moduli is often a very polemical subject. Current practice is such that the use of 1024-bit moduli is usually considered to be sufficient to guarantee practical security, even if analyses carried out by the greatest specialists in the subject agree on the concept that such moduli do not provide sufficient security, or at least security comparable to what is demanded from other cryptographic mechanisms. Application of a fundamental paradigm of cryptography, that consists in calibrating systems imposing a safety margin rather than designing them to be at the limit of known attack capabilities (see G.1.2, page 57), argues for the use of moduli containing at least 1536 bits, although no 1024-bit RSA modulus has been officially factorized at the moment[5]. Consequently, in our opinion the use of 1024-bit moduli would mean taking too high a risk regarding the criteria for the standard cryptographic robustness level.
- Appendices G.1.2 and G.2.2 provide further information about analyses related to the factorization problem.
- The use of particular secret exponents to improve performances should be avoided due to practical attacks published on this subject.
- The use of very small public exponents, such as an exponent equal to 3, should also be avoided for encryption due to existing attacks. More generally, the use of such exponents is not recommended for any application, for security reasons.
- The use of prime numbers p and q too close from one another or with excessively different sizes can compromise security. It is also important to avoid values with particular properties such as the lack of a large prime factor in the decomposition of p-1 or q-1. It is recommended that p and q should be chosen at random among prime numbers with a size equal to half the size of the modulus, to prevent these problems.

C.2.1.2. Discrete logarithm in GF(p)

The so-called "discrete logarithm in GF(p)" problem is based on computations carried out in the prime Galois Field with *p* elements, where *p* is a prime number.

Standard level

Rule$_S$Logp-1. The minimum size of prime moduli for use not beyond the year 2010 is 1536 bits.

Rule$_S$Logp-2. The minimum size of prime moduli for use not beyond the year 2020 is 2048 bits.

Rule$_S$Logp-3. The minimum size of prime moduli for use after 2020 is 4096 bits.

Rule$_S$Logp-4. For use not beyond 2010, the order of sub-groups used shall be equal to a multiple of a prime number with at least 160 bits.

Rule$_S$Logp-5. For use beyond 2010, the order of sub-groups used shall be equal to a multiple of a prime number with at least 256 bits.

[5] A recent published result announces the factorization of a 1039-bit number. However this number is not a RSA modulus.

Recom$_S$Logp-1. For use not beyond 2010, it is recommended that the prime moduli used should contain at least 2048 bits.

Recom$_S$Logp-2. Even for use not beyond 2010, it is recommended that the order of sub-groups used should be a multiple of a prime number with at least 256 bits.

Justification:

- The complexity of the discrete logarithm problem appears to be comparable to the complexity of factorization. Similar methods can be applied to solve both problems and one may reasonably believe that major progress towards solving the factorization problem will be accompanied by similar progress in the discrete logarithm problem. Consequently, it is natural to apply identical rules for the two problems.
- However, the discrete logarithm problem appears slightly more difficult in practice, because there is a difference of between about 100 and 200 bits in computation records (see G.1.2) for the two methods, due to the fact that some computation phases are more difficult for the discrete logarithm problem than for the factorization problem, but the question arises whether such a difference might be largely due to the prestige accompanying a record in factorization.

C.2.1.3. Discrete logarithm in GF(2^n)

The so-called "discrete logarithm in GF(2^n)" problem is based on computations carried out in the Galois Field with 2^n elements, where n is an integer.

Standard level

Rule$_S$Log2-1. The minimum value of *n* for use not beyond 2020 is 2048 bits.

Rule$_S$Log2-2. The minimum value of *n* for use beyond 2020 is 4096 bits.

Rule$_S$Log2-3. For use not beyond 2010, the order of sub-groups used shall be equal to a multiple of a prime number containing at least 160 bits.

Rule$_S$Log2-4. For use beyond 2010, the order of sub-groups used shall be equal to a multiple of a prime number containing at least 256 bits.

Recom$_S$Log2-1. It is recommended that public key primitives that are not based on the discrete logarithm in GF (2^n) problem should be preferred.

Justification:

- Although it is difficult, the discrete logarithm in GF (2^n) problem appears to be easier than in GF(p), for equal (Galois) Field size. The use of GF(2^n) instead of GF(p) is only justified in many applications by computational efficiency reasons. Consequently, for security reasons we recommend that the use of GF(p) should be preferred.

C.2.1.4. Elliptic curves defined in GF(p)

It is also possible to define discrete logarithm problems in more complex structures for which no known algorithm is more efficient than generic discrete logarithm computational methods. This is particularly the case at the moment for elliptic curves that are defined on a basic (Galois) Field which in practice can be prime (GF(p)) or binary (GF(2^n)).

Standard level

Rule$_S$ECp-1. For use not beyond 2010, the order of sub-groups used shall be equal to a multiple of a prime number containing at least 160 bits.

Rule$_S$ECp-2. For use beyond 2010, the order of sub-groups used shall be equal to a multiple of a prime number containing at least 256 bits.

Rule$_S$ECp-3. If particular curves are to be used in which security is based on a mathematical problem easier than the generic discrete logarithm on an elliptic curve defined in GF(p), this problem shall comply with the corresponding rules for the standard level.

Justification:

- The discrete logarithm problem on elliptic curves appears to be a very difficult problem at the moment. Already for small parameter sizes the security that it provides is comparable to the security required for symmetrical primitives.
- Security can be seriously degraded if curves generally described as being “particular” are used. Notably, the underlying mathematical problem may reduce to a discrete logarithm problem in a Galois field and no longer on an elliptic curve. In this case, rules related to this problem are obviously applicable.

Standard level mechanism(s):

- The use of curves P-256, P-384 and P-521 defined in FIPS 186-2 date 27/01/2000 is compatible with the standard level.

C.2.1.5. Elliptic curves defined in GF(2^n)

Standard level

Rule$_S$EC2-1. For use not beyond 2010, the order of sub-groups shall be equal to a multiple of a prime number containing at least 160 bits.

Rule$_S$EC2-2. For use beyond 2010, the order of sub-groups shall be equal to a multiple of a prime number containing at least 256 bits.

Rule$_S$EC2-3. The parameter *n* must be a prime number.

Rule$_S$EC2-4. If particular curves are to be used in which security is based on a mathematical problem easier than the generic discrete logarithm on an elliptic curve defined in GF(2^n), this problem shall comply with the corresponding rules for the standard level.

Justification:

- The use of composite *n* considerably reduces the difficulty of the discrete logarithm and therefore weakens the corresponding mechanism.
- In terms of standard robustness, we do not make any differentiation between the elliptic curves defined on GF(p) and the elliptic curves defined on GF(2^n).

Standard level mechanism(s):

- The use of curves B-283, B-409 and B-571 defined in FIPS 186-2 dated 27/01/2000 is compatible with the standard level.

C.2.1.6. Other problems

Several other problems have been proposed in the academic environment to provide alternatives to the problems mentioned above. Since these problems have been only used very infrequently in practice, their security should be considered case by case.

C.2.2. Asymmetric encryption

Asymmetric encryption methods are based on difficult basic problems. Therefore, this basic problem must comply with the required robustness level. It is also possible for some encryption mechanisms to demonstrate that their security is equivalent to the basic problem and is not simply related to it heuristically, possibly making some assumptions. This modern cryptographic approach can give a better degree of assurance than a simple approach stating that there are no known attacks.

Standard level

Recom$_S$AsymEnc-1. It is recommended that asymmetric encryption mechanisms with proven security should be used.

Standard level mechanism(s):

- The RSAES-OAEP asymmetric encryption mechanism defined in document PKCS#1 v2.1 is standard level provided that rules **Rule$_S$Fact-1**, **Rule$_S$Fact-2**, **Rule$_S$Fact-3, Rule$_S$Fact-4** and **Rule$_S$Fact-5** are respected.

C.2.3. Digital signature

All asymmetric signature methods are based on a difficult basic problem. Therefore, this must match the required degree of robustness. It is also possible for some encryption mechanisms to demonstrate that their security is equivalent to the basic problem and is not simply related to it heuristically, possibly making some assumptions.

Signature schemes also generally use hash functions for which the robustness level (see C.3.1.) must match the required robustness level for the signature mechanism.

Standard level

Recom$_S$AsymSign-1. It is recommended that asymmetric signature mechanisms with proven security should be used.

Standard level mechanism(s):

- The RSASSA-PSS[6] asymmetric signature mechanism defined in document PKCS#1 v2.1 is standard level provided that rules **Rule$_S$Fact-1**, **Rule$_S$Fact-2**, **Rule$_S$Fact-3** and **Rule$_S$Fact-4** are respected.
- The ECDSA asymmetric signature mechanism based on elliptic curves defined in FIPS 186-2 is standard level if it uses one of the P-256, P-384, P-521, B-283, B-409 or B-571 curves.

C.2.4. Asymmetrical authentication of entities and key exchange

As mentioned in F.2.3, interactive entity authentication mechanisms and key exchange mechanisms are generally based on random number generation, hashing and public key encryption or signature mechanisms. Therefore, the rules mentioned in sections C.2.2, C.2.3, C.3.1 and C.3.2 are directly applicable.

C.3. Other cryptographic primitives

C.3.1. Hash function

Cryptographic hash functions must have several properties such as the lack of "collisions" (see F.3.1). However, such collisions can always be found by generic attacks called "birthday paradox attacks". Consequently, one of the purposes when designing a hash function is to assure that no better attack exists. In practice, in order to prevent practical "birthday paradox attacks" hash sizes must have twice the size of a symmetric key to achieve the same degree of robustness.

Furthermore, hash functions are usually built up around more elementary functions called compression functions. The existence of attacks on these components described below as being "partial attacks", does not necessarily suggest that it is possible to attack the hash function in itself, but it demonstrates major design flaws.

Standard level

Rule$_S$Hash-1. For use not beyond 2010, the minimum hash size generated by a hash function is 160 bits.

Rule$_S$Hash-2. For use beyond 2010, the minimum hash size generated by a hash function is 256 bits.

Rule$_S$Hash-3. The best known attack to find collisions must require of the order of $2^{h/2}$ hash computations, where *h* is the hash size in bits.

[6] RSASSA-PSS: "RSA Signature Scheme with Appendix" – Provably Secure encoding method for digital Signatures".

Recom$_S$Hash-1. The use of hash functions for which "partial attacks" are known is not recommended.

Justification:

- Rules for hash sizes are directly derived from rules applied in symmetric cryptography.
- We would like to emphasize the fact that the existence of partial attacks, even if they do not lead to an attack on the hash function, is a sign of serious design flaws.
- The security criterion used to judge the quality of a hash function is the absence of known collisions. In some applications, this property appears too strong because all that must be impossible is the "pre-image" computation. However, we consider that regardless of any context, the only way to assign a hash function to a given robustness level is the collision resistance property. This is not contradictory with the possibility that a mechanism using a hash function that is not at the standard level can be considered as globally reaching the standard level.

Standard level mechanism:

- The SHA-256 hashing mechanism defined in FIPS 180-2 is standard level.

Mechanism not reaching the standard level:

- A collision attack against the SHA-1 hashing mechanism defined in the FIPS 180-2 has recently been published. The complexity of this attack is estimated at 2^{63}, in other words less than 2^{80}. Although this attack has not led to the computation of an explicit collision at the time that this document is written, its simple existence shows that there is a serious weakness in the security of this function. Therefore, the SHA-1 hashing mechanism does not reach the standard level.

C.3.2. Generation of cryptographic random numbers

As explained in F.3.2, the generation of random numbers usually consists of a combination of a physical process with at least partially random behavior, and an "**algorithmic post-processing**" step using a pseudo-random generator. An analysis of the physical process goes outside the scope of cryptography. However, the algorithmic post-processing analysis can be made by methods conventionally used in cryptography, taking account of assumptions about the "quality" of the physical random numbers.

In the following, a distinction is made between two types of physical sources. The first collects elements from the environment of the mechanism with variations that could be partially random (internal clock in a PC, mouse movements, keyboard typing intervals, equipment interrupt signals, etc.). This type of source is called "**external elements**". It is not considered as being a genuine continuous random number source, and therefore does not directly participate in security of the random number generator. This is why it is not shown in

Figure 1. However, it would be possible to use data derived from this type of source as an initial seed in a pseudo-random generator. The second type of physical source consists of physical mechanisms specially designed to generate random numbers throughout the life of the system. This type of source is called a "**true random number generator**" and it must comply with some rules and recommendations, given under C.3.2.2.

Combined with a true random number generator, it is possible and even recommended that "**secret elements**" like secret keys specific to each equipment, and therefore not shared, should be used. Finally, the use of an accumulator consisting of a "**non-volatile memory**" may be very useful to prevent any form of resetting of the generator.

The following architecture is then obtained, in which not all components are always necessary for each target robustness level:

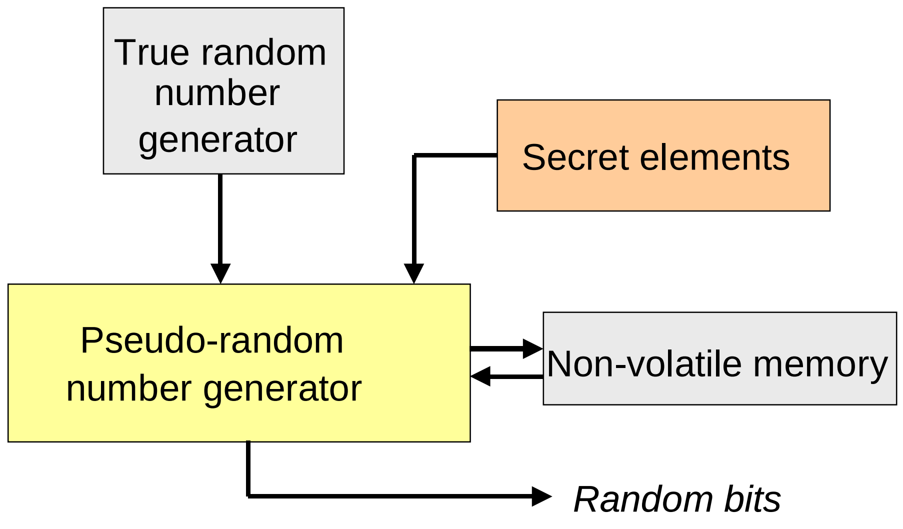


Figure 1. Generic architecture of a random number generator

Rules and recommendations for the random number generator are based on the observation that at the present time it is very difficult to provide convincing proof about the quality of the random numbers generated by a physical generator, although it is relatively easy to be convinced of the quality of good post-processing. However, these rules may be revised if major theoretical progress is made in the field of true random number generators.

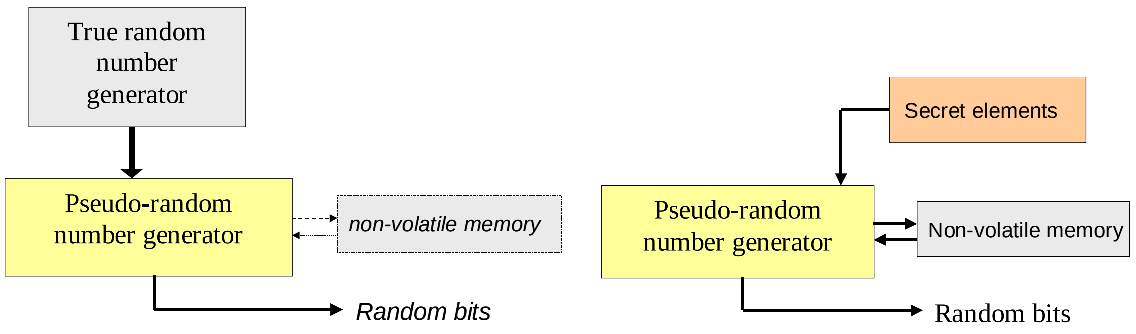
C.3.2.1. Architecture of a random number generator

Standard level

Rule$_S$ArchiRNG-1. A pseudo-random algorithmic post-processing must be used.

Rule$_S$ArchiRNG-2. A true random number generator <u>or</u> secret elements combined with a non-volatile memory must be used.

Recom$_S$ArchiRNG-1. In all cases, the use of a non-volatile memory as an accumulator is recommended.


True random number generator
Pseudo-random number generator
non-volatile memory
Random bits
Secret elements
Pseudo-random number generator
Non-volatile memory
Random bits

<u>Justification:</u>

- The direct use of a true random number generator is not accepted. Similarly, the use of elementary post-processing, frequently described as “smoothing”, is not accepted.

<u>Comments:</u>

- A pseudo-random generator is inherently very different from a true random number generator. The post-processing is a deterministic algorithm that does not generate random numbers, but simply generates sequences of bits indistinguishable from genuinely random sequences starting from a small random “germ”. In particular, without the initial random part provided by the germ, which may for example form the secret elements mentioned above, the behavior of the reprocessing algorithm would be perfectly predictable.
- It is easy to control correct operation of algorithmic post-processing, like any other deterministic cryptographic mechanism. There is a major difference with true random number generators for which it is only possible to check that they are not in an obvious failure state by means of statistical tests. In particular, the use of statistical failure tests at the output of an algorithmic post-processing is not only pointless, but it can even be dangerous for security. Furthermore, the same rules should be used for implementation of post-processing algorithms as for any other cryptographic mechanism.

<u>Standard level mechanism(s):</u>

- The random number cryptographic post-processing mechanism defined in standard ANSI X9.31 and described in page 52 is standard level if it is used with the AES as the block cipher mechanism.

C.3.2.2. True random number generator

The true random number generator is designed to generate random numbers throughout the life of the system. Therefore, the quality and reliability of these random numbers need to be guaranteed with certainty.

Standard level

Rule$_S$TRNG-1. The true random number generator must have a functional description. In particular, the description must state the principles on which generation of the genuine random numbers is based.

Rule$_S$TRNG-2. Statistical tests at the output from the true random number generator must not show any defects in the generated random numbers.

Recom$_S$TRNG-1. It is desirable that the quality of the random numbers produced by the true random number generator should be justified by reasoning.

Justification:

- The design of a true random number generator is not based simply on a combination of physical components, hoping that the assembly obtained will provide satisfactory randomness. A careful study must guide the designer in his choices. Therefore these design choices and the study that guided them must appear in a document. Among other things, the purpose is to describe the physical sources used and the processing applied to these sources.
- It is often recommended that the quality of the random numbers derived from a physical source should be monitored by performing elementary statistical tests in order to detect any blockages. Obviously, these tests must be carried out before any post-processing. It is also important to very specifically specify the behavior to be followed if these tests detect any failures.
- The next step, that is a simple recommendation at the standard level, is to justify choices made by either heuristic or rigorous, qualitative or quantified reasoning. The form and type of reasoning are left free. Its purpose is to convince designers and users that the true random number generator actually produces genuine random numbers.
- It is quite clear at the moment that it is very difficult to provide convincing proof about the quality of the random numbers produced by a physical generator. Therefore, it is necessary to perform statistical tests on a representative sample derived directly from the true random number generator. These tests are used for a *posteriori* validation of design choices. For example, tests recommended by the NIST (FIPS 140-2 and SP 800-22) could be used, but any tests that appear to be relevant can also be used.

Comments:

- Statistical tests covered by the rule Rule$_S$TRNG-2 are first-time tests or occasional tests. They are not failure tests to be carried out during operation, while the generator is in use.

C.3.2.3. Pseudo-random number generator

Standard level

Rule$_S$PRNG-1. The pseudo-random number generator used must be standard level.

Justification:

- To be standard level, the cryptographic pseudo-random post-processing mechanism used must use algorithms with at least standard robustness level (hash function, block cipher, etc.).

C.3.3. Management of keys

Management of keys is a quite separate problem that is sometimes just as complex as determination of encryption and integrity algorithms. Therefore, this problem is dealt with in a document dedicated to it. This document is entitled “Management of cryptographic keys – Rules and recommendations for management of keys used in cryptographic mechanisms” (currently available only in French). It is also broken down into different versions following the same distribution principles as this document. A few rules are given in the remainder of this section to give an initial overview of the key management problem. These rules are not exhaustive and the reader can refer to the dedicated document to determine the extent of rules governing management of keys.

Before dealing with rules and recommendations about the management of keys, note that key escrow is a quite separate mechanism for which the robustness level must be estimated using the same rules as for any other cryptographic mechanism. This problem is also dealt with in the “Management of cryptographic keys” document.

C.3.3.1. Symmetric secret keys

The main two general risks in the use of secret keys are firstly use of a key for several purposes (for example in confidentiality and integrity) and secondly the use of keys shared by a large number of users or equipments (see F.3.3.1). Such keys are referred to herein using the term “**common keys**”, in the sense that they are held by a large number of players at the same hierarchical level within a system. These keys are also called “network keys” in some applications.

Common keys must not be confused with “**master keys**” used to generate “**derived keys**”. In this case, only derived keys are used in products, and not the master keys used to generate them. This should also be distinguished from the case of “**differentiated keys**” that are generated locally from a same secret key, for use by distinct mechanisms.

Standard level

Rule$_S$GestSym-1. A single key shall not be used for more than one purpose.

Rule$_S$GestSym-2. If differentiated keys are used in a standard robustness level mechanism, these keys must be generated using a standard level diversification mechanism.

Rule$_S$GestSym-3. If there are any derived keys, they must be generated using a “coherent” level diversification mechanism.

Recom$_S$GestSym-1. The use of common keys is not recommended.

Justification:

- The use of the same key for more than one purpose, for example to encrypt with a confidentiality mechanism and to guarantee integrity with a different mechanism, is the source of many errors. However, this does not mean that two keys cannot be differentiated locally starting from the same secret key, provided that the diversification mechanism has the standard cryptographic robustness level.
- Rule **Rule$_S$GestSym-3** indicates the need to calibrate the system such that preferred attack targets such as the master keys are sufficiently well protected.
- The use of common keys is not recommended because such keys are preferred attack targets.

C.3.3.2. Asymmetric key pairs

A key management infrastructure is usually made hierarchically, each public key being certified by a key with the immediately higher rank until reaching a root key.

Standard level

Rule$_S$GestAsym-1. The use of the same key pair for more than one purpose is not accepted.

Rule$_S$GestAsym-2. Hierarchically important keys such as root keys must be generated and used by “coherent” level mechanisms.

Justification:

- The use of the same key pair for more than one use, for example to encrypt and to sign, is a source of serious errors.
- The **Rule$_S$GestAsym-2** rule indicates the need to calibrate the system such that preferred attack targets such as root keys are sufficiently robust.

D. Bibliography

[1] W. Diffie and M. Hellman. *New directions in cryptography*. IEEE Transactions on Information Theory, 22 (1976), 644-654.

[2] Arjen Lenstra and Eric Verheul. *Selecting Cryptographic key Sizes*. Journal of Cryptology, volume 14, number 4, pp. 255-293, Springer – Verlag, December 2001.

[3] Alfred Menezes, Paul J. Oorschot and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press, 1997. (**http://www.cacr.math.uwaterloo.ca/hac**)

[4] Bruce Schneier. *Cryptographie appliquée (Applied Cryptography)*. Vuibert, 2001.

[5] Simon Singh. *Histoire des codes secrets (Secret code history)*. JC Lattès, 1999.

[6] Jacques Stern. *La science du secret (The science of privacy)*. Editions Odile Jacob, 1998.

[7] Douglas Stinson. *Cryptographie, théorie et pratique (Cryptography, theory and practice)*. Vuibert, 2001.

E. Acronyms, abbreviations and definitions

	Definition	Page(s)
AES	Advanced Encryption Standard	15
ANSI	American National Standard Institute	
AsTeC	Cellule d’Assistance Technique en Conception de la DCSSI (DCSSI Technical Assistance Unit for Systems Design)	
CBC	Cipher-Block Chaining mode of operation	16
CBC-MAC	Cipher-Block Chaining Message Authentication Code	19
CCA	Chosen Ciphertext Attack	16
CFB	Cipher Feedback mode of operation	16
CTR	Counter mode of operation	
DCSSI	Direction Centrale de la Sécurité des Systèmes d’Information (Central Division for Security of Information Systems)	
DES	Data Encryption Standard	15
DFA	Differential Fault Analysis	7
DPA	Differential Power Analysis	7
DSA	Digital Signature Algorithm	26
ECB	Electronic Codebook mode of operation	16
ECDSA	Elliptic Curve Digital Signature Algorithm	26
FIPS	Federal Information Processing Standard (NIST standard)	
GF(2^n)	Binary Galois Field with 2^n elements	23, 25
GF (p)	Prime Galois Field with *p* elements	23, 24
HODPA	High Order Definition Power Analysis	7
MAC	Message Authentication Code	19
NIST	National Institute of Standards and Technology	
OAEP	Optimal Asymmetric Encryption Padding	25
OFB	Output Feedback mode of operation	16
PIN	Personal Identification Number	39
PKCS	Public Key Cryptography Standards	
PSS	Provably Secure encoding method for digital Signatures	26
RSA	“Rivest-Shamir-Adleman” (asymmetric cryptography)	25, 26
SGDN	Secrétariat Général de la Defense National (National Defence General Secretariat)	
SHA	Secure Hash Algorithm	27
SPA	Simple Power Analysis	7

F. Definitions and concepts

1 The purpose of this chapter is to give reminders of essential definitions and concepts in cryptography, so as to clarify rules and recommendations issued in this document. These reminders cover the strict minimum. Obviously they are brief, and deliberately non-mathematical. Further information can be obtained for example, in the excellent reference book [3]. A great deal of information can also be found in [6], [4] and [7]. Document [5] gives a most historical approach.

2 Additional information that can clarify some concepts but is not necessary in a first reading is described in small characters, in the same style as this paragraph.

3 **Cryptology** is a science at the boundary between mathematics and computer science, and is traditionally defined as being the "science of secrets". For many years, it concentrated on the confidentiality problem, essentially for military or diplomatic purposes. Recently, cryptology entered a new era under the double influence of large theoretical and technical progress, and secondly the emergence of new needs from what is commonly called the information society.

4 Cryptology deals with the design, security and use of cryptographic mechanisms. These mechanisms are often described as being cryptographic **primitives**, to emphasize their elementary indivisible nature, but also the need to assemble them in order to obtain complete systems capable of providing security services. The term cryptographic **algorithms** is also used to indicate that the target description precision must be sufficient to implement them.

5 Traditionally, cryptology is divided into two branches, depending on the point of view (designer's or attacker's); **cryptography** studies the design of mechanisms to achieve confidentiality, integrity or authenticity of information and technically similar mechanisms such as identification of entities. **Cryptanalysis** is concerned with the same primitives, and attempts to analyze their security or even to break it. Obviously, there can be no cryptography without cryptanalysis and vice versa.

6 Another way of separating cryptology into two branches, which is orthogonal to the above, is more technical in nature. A distinction is usually made between so-called **symmetric** cryptography, also described as **conventional** or **secret key** cryptography, and **asymmetric** cryptography or **public key** cryptography. This separation arises from the founding article by W. Diffie and M. Hellman [1] that was a major breakthrough in cryptology in 1976, notably by suggesting that a message to a given addressee can be encrypted without necessarily sharing a secret key with him in advance. A new approach to cryptology, described as being asymmetric, was born. Since then, primitives have often been classified depending on their nature (symmetric or asymmetric). In the remainder of this chapter we have decided to use this first separation to summarize the different cryptographic primitives.

7 A second dimension for classification of cryptographic primitives is based on the target security goal. Traditionally, a distinction is made between confidentiality and/or data integrity, data authentication or entity authentication needs, and non-repudiation needs. Obviously, other functions are possible but the functions that we have just mentioned are by far the most important functions dealt with by cryptographic means.

8 The purpose of **confidentiality** is to assure that transmitted or stored information is only accessible to persons authorized to receive it. This security objective is conventionally assured by encryption, but obviously any other appropriate means can be used, beginning with non-cryptographic organizational methods.

9 Assuring data **integrity** consists in preventing any unauthorized corruption of data, or at least detecting such corruption if it occurs. Corruption means any modification, partial deletion or insertion of information. Data integrity is conventionally assured in cryptography by message authentication codes or digital signature mechanisms.

10 **Data authentication** aims at assuring that the data are actually sent by a particular source. Such an authentication confirms integrity of this information and is made by the mechanisms mentioned above. **Entity authentication**, also referred to as **identification**, is intended to assure that a person is actually who he/she claims to be. It may be done in various symmetric or asymmetric ways.

11 **Non-repudiation** is a service designed to prevent an entity from denying having performed a given action. The main non-repudiation mechanism is signature, a signed message usually remaining signed even if the signatory subsequently changes his opinion.

12 We now consider the main primitives available in modern cryptography. The following table aims to classify them depending on their symmetric or asymmetric nature, and their security objective. Obviously, other very interesting primitives could be mentioned, but they do not naturally appear in such a table. The remainder of this chapter briefly describes these primitives and summarizes the essential points necessary for understanding the remainder of this document.

		Symmetric cryptography	Asymmetric cryptography
Confidentiality		Conventional encryption by block cipher (F.1.1.1.) or stream cipher (F.1.1.2.)	Public key encryption (F.2.1)
			Key exchange (F.2.3.)
Integrity		Message authentication code (F.1.2)	Digital signature (F.2.2, F.2.3)
Authentication	of data		
	of entities	Challenge - response (F.1.3.)	
Non-repudiation			

F.1. Symmetric cryptography

13 Symmetric cryptographic primitives are characterized by the fact that they use cryptographic keys shared by several persons or entities. Consequently, a **symmetric secret key** is simply a secret element. Therefore the security of a system using such keys is based particularly on protection of these secrets.

14 A secret key is usually an arbitrary sequence of bits, i.e. 0s and 1s, with fixed size. This key size, denoted *n* in this document, is of overriding importance for security of the system because exactly 2^n keys with length *n* can be formed. Symmetric systems are usually vulnerable generically by enumerating all possible keys. Such an attack called an "exhaustive search" can only be successful if the number of computations appears to be feasible using reasonable computer means. Therefore, the ability to carry out 2^n operations provides a lower limit to the size of symmetric systems.

15 In order to give an indication about orders of magnitude handled and especially to realize that 2^n is very quickly a gigantic number as *n* increases, we have included a few specific numeric examples in the following table.

n	2^n
32	$2^{32} \approx$ number of humans on earth
46	$2^{46} \approx$ distance Earth – Sun in millimeters $2^{46} \approx$ number of operations carried out in a day, at a billion operations per second (1 GHz)
55	$2^{55} \approx$ number of operations carried out in a year, at a billion operations per second (1 GHz)
82	$2^{82} \approx$ mass of the Earth in kilograms
90	$2^{90} \approx$ number of operations carried out in 15 billion years (age of the universe) at a billion operations per second (1GHz)
155	$2^{155} \approx$ number of water molecules on earth
256	$2^{256} \approx$ number of electrons in the universe

16 These figures simply show that the function 2^n increases extremely quickly with *n*. Therefore, the ability to carry out 2^{128} computations appears very unlikely in our lifetime, even with very large means. In any case, to solve any doubts, it would appear that the ability to perform 2^{256} computations is absolutely impossible and always will be. This is one of the rare certainties that can be obtained in cryptography. In particular, this goes against the common idea by which every cryptosystem can necessarily be broken by an exhaustive search provided that sufficient means are available.

F.1.1. Symmetric encryption

17 The best known symmetric primitives are encryption algorithms. They use secret keys known only to message senders and receivers, to protect confidentiality of this information, even if the communication channel used is being listened to. However, it must be clear that this does not solve the major problem of the initial exchange of secret keys between the correspondents.

18 Encryption algorithms can also be used to store information securely without any intention of transmitting it. Thus, information recorded on potentially vulnerable supports can be kept confidential.

19 However, protecting the confidentiality of information only guarantees that the content of this information is inaccessible to an attacker. On the other hand, there is no guarantee about the integrity or authenticity provided by encryption methods, and nothing makes it impossible to perform active attacks so as to modify transmitted or stored information. Thus, it is easy to imagine attack scenarios during which an attacker can modify confidentially protected communications, without needing to understand precisely what is being transmitted. Section 39 provides a specific example of such an attack.

20 Symmetric encryption methods, also called "conventional encryption" or "secret key encryption" methods are naturally divided into two families, namely **block cipher** and **stream cipher,** described below.

F.1.1.1. Block cipher

21 A block cipher primitive is an algorithm that processes data to be encrypted by fixed-size blocks. The number of bits in these data blocks will be denoted k; typically this size is equal to 64 or 128 bits in practice. Therefore, such a mechanism can only be used to combine a sequence of k data bits with a key of n bits, in order to obtain an encrypted data block with the same size k as the plain data block.

22 The main expected property of a block cipher mechanism is to be easily inverted if the secret encryption key is known — this is then referred to as decryption — but to be impossible to invert in practice when this information is not available. Only holders of the secret key are capable of transforming plain data into encrypted data and, conversely, to transform the encrypted data into plain data.

23 The most well-known example of a block cipher algorithm is the DES (*Data Encryption Standard*) defined in 1977 by the NIST, i.e. the American National Institute of Standards and Technology, as being the encryption standard for commercial applications. It processes $k = 64$ bit blocks using $n = 56$ bit keys. Much ink has been spilt about the security of the DES. However, this algorithm was particularly well designed, because the best known practical attack was an exhaustive search on the keys. However, the keys are undersized, which makes this attack realistic at the moment, using a dedicated machine, in only a few hours.

24 However, the DES is still frequently used, but in the form of the "triple-DES", a variant using 112-bit keys that cannot be attacked by an exhaustive search. The medium term objective is to replace it by the AES (Advanced Encryption Standard) selected by the NIST in an international competition. The AES is designed to process 128 bit blocks using 128, 192 or 256-bit keys.

25 More generally, a block cipher algorithm combines **substitution** operations designed to replace symbols by other symbols in order to conceal the meaning, with **permutation** operations that exchange the position of symbols. These two principles are historically very old but are still valid at the moment, since any block cipher primitive can be seen as being an intelligent combination of these two operations.

26 In this respect, it should be understood precisely what a block cipher algorithm does, and especially what it does not do. Firstly, such an algorithm can only process relatively small fixed size blocks. Consequently, if arbitrary sized messages are to be encrypted, it is necessary to define how the message should be encoded into a sequence of fixed size blocks. This requires that a very precise definition should be made of how to distribute the information in such messages into a sequence of bits with length equal to an exact multiple of the size k of the elementary block. This operation is called padding.

27 Furthermore, a block cipher algorithm is basically deterministic. It always produces the same encrypted block if it starts from a given data block and a secret key. Thus, a passive attacker observing two identical encrypted blocks can immediately deduce that the corresponding plaintext blocks are also identical, although without obtaining any information about the content of the plaintext. However, under some circumstances such information may be sufficient to attack a system.

28 As an example, imagine a bank system in which a four-digit PIN code (Personal Identification Number) for a credit card can be verified by encrypting this code and sending it to a bank host computer. If a simple block cipher is used with a fixed bank key, a single encrypted value will always be used for each PIN code. It would then be possible, for example, to imagine that an attacker observing the communications will immediately learn who has the same PIN code as himself.

29 Therefore, in order to process messages of an arbitrary size and to assure global confidentiality of these messages and not simply confidentiality "by block", a **mode of operation** should be defined. This mode of operation should specify the initial processing of the message into a sequence of blocks and how these blocks are encrypted so as to obtain the encrypted message at the end. Note that the definition of such a procedure does not need to take account of the details of the underlying block cipher algorithm; all that is really necessary is the size k of the blocks. Note also that in order to break the deterministic nature of the encryption, the process needs to be "randomized" i.e. a certain amount of a random variability needs to be introduced.

30 The most well-known mode of operation is CBC (Cipher-Block Chaining). One of many variants works as follows; the starting point for encrypting a message formed from a sequence of bits is to add a bit equal to 1 and as many bits equal to 0 as necessary to obtain a total number of bits equal to a multiple of the block size. We will denote the t plaintext blocks thus obtained as $X_1, X_2, \ldots X_t$. The next step is to choose a random block denoted IV

as the "*initial value*". This block must be independent of the message and the previous encryptions. If the result of the encryption of block x using key K is denoted $E_k(x)$, the encrypted form $(c_0, c_1, c_2, \ldots, c_t)$ of the message is obtained by setting $c_0 = IV$ and then calculating $c_i = E_K(c_i\text{-}1 \oplus x_i)$ in sequence for all values of i varying from 1 to t (the "exclusive or" operator denoted "⊕" represents the bit by bit addition without any retainer). Graphically, the schematic representation shown in Figure 2 is obtained.

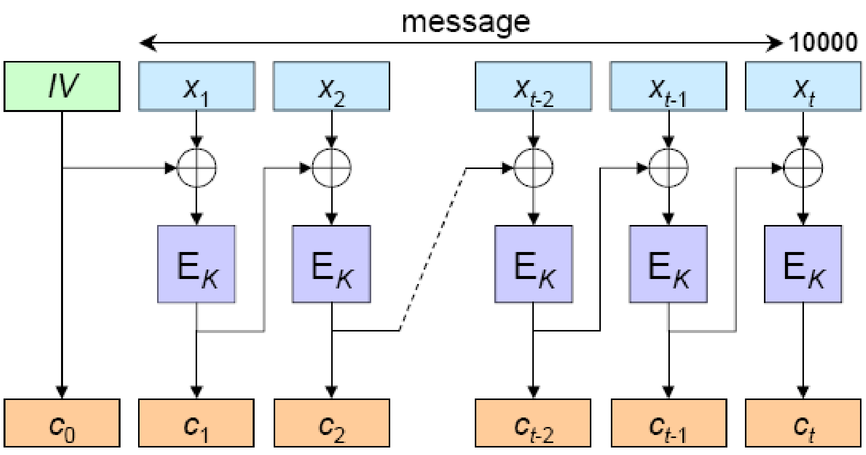


Figure 2: CBC mode of operation

31 This mode of operation illustrates the initial "*padding*" phase of completing the message by a bit equal to 1 followed by zero bits. It also shows that use of random data through the IV can make the encryption non-deterministic. Thus, encryption of the same message twice consecutively will have only an infinitely small chance of using the same IV and consequently the ciphertext will be different due to the feedback mechanism using the encrypted block c_{i-1} during encryption of the plaintext block x_i. Note also that in this variant of the CBC mode, the IV is transmitted in plain text without needing to be encrypted.

32 In order to decrypt a message $(c_0, c_1, c_2, \ldots, c_t)$, $x_i = c_{i-1} \oplus D_K(c_i)$ has to be computed for i varying from 1 to t, $D_K(c)$ denoting the decrypted content of block c with secret key K. It is easy to check that the message thus obtained is actually the initial message. Finally, note that this mode naturally has a self-synchronization property; all that is necessary to correctly resume decryption if encrypted blocks are lost during transmission, is to obtain two successive intact blocks.

F.1.1.2. Stream cipher

33 Stream cipher primitives use a different approach from block cipher, in that they usually consider the message to be encrypted as a sequence of bits that are combined with a sequence of bits derived from the secret key.

34 Stream cipher algorithms are based on the Vernam encryption that is very simple, reliable and unusable in practice. Considering a message represented in the form of a sequence of bits $m_1, m_2, m_3, \ldots$ and a key also seen as a sequence of bits $k_1, k_2, k_3, \ldots$, the encrypted content of the message is then very easily obtained using the operation of "bitwise exclusive or" in which the i^{th} encrypted bit c_i is obtained by addition (without carry) of the i^{th} message bit with the i^{th} key bit, namely $c_i = m \oplus k_i$. Thus, $0 \oplus 0=0$, $0 \oplus 1 = 1 \oplus 0 = 1$ and $1 \oplus 1 = 0$, and this last operation is the only operation different from a classical addition.

35 Decryption is then done simply by applying the same bit by bit addition operation of the encrypted message with the secret key because $c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i = m_i \oplus (k_i \oplus k_i) = m_i \oplus 0 = m_i$, noting that regardless of the value k_i of each key bit, $k_i \oplus k_i$ is always equal to 0.

36 It is easy to demonstrate that the Vernam encryption is perfectly reliable in terms of confidentiality, provided that the key is at least as large as the message to be encrypted and is only used once. Obviously, this restricts possible applications due to the size of the key to be shared between the sender and the receiver. In most cases, agreement about the secret key causes a problem similar to secure transmission of the message itself.

37 It is fairly easy to demonstrate that the Vernam cryptosystem is the only encryption mechanism capable of giving perfect confidentiality of data. Claude Shannon had come to the conclusion that it was impossible to design such perfect cryptographic primitives that could be used in practice as early as 1949. The modern approach does not contradict this concept but it accepts imperfect security; the entire art of the cryptographer is to estimate this degree of imperfection and to imagine primitives for which it can be made negligible. Therefore, no attempt is made to protect against an adversary with an infinite computational power, but rather against a computational power for which the upper limit can be estimated.

38 The key idea of stream ciphers is to use small keys, typically of the order of 128 bits, and to deterministically —thus perfectly reproducibly— derive random sequences that can be used as keys of the same length as the message for the Vernam encryption algorithm[7]. Decryption then acts in the same way by generating the same sequence from the secret key.

39 Similarly to the warning in section 19, stream ciphers, derived from the Vernam algorithm, provide a simple and particularly eloquent example of the fact that confidentiality, even perfect, does not automatically imply the integrity of the transmitted message. If an attacker wishes to invert a bit of a plain message, i.e. to transform a 0 into 1 or vice versa, all that he needs to do is to add 1 to the encrypted bit c_i, and therefore to transmit $c'_i = c_i \oplus 1$. During decryption, the receiver will calculate $c'_i \oplus k_i = c_i \oplus 1 \oplus k_i = m_i \oplus 1$; if m_i is equal to 0, the result obtained will be 1 and vice versa if m_i is equal to 1 the result will be $1 \oplus 1 = 0$. Consequently, even if the attacker has no idea of the value of the bit being modified, he can certainly and effortlessly invert it. As an application example, it would be possible to imagine encryption of the amount of a financial transaction; the amount to be paid is usually defined at a fixed and easily determined position in the message. A bit in this amount could then be inverted and thus a small amount could easily be transformed into a large amount, for example by transforming 0000017 Euros into 1000017 Euros in decimal notation.

40 The conventional confusion between confidentiality and integrity is often strengthened by excessively simplistic images of the action mechanism of cryptographic primitives. Thus, encryption is often presented as being the electronic version of an opaque envelope or worst, a safe. Since it is difficult to imagine modifying information contained in a safe without becoming aware of it, it would be easy but wrong to conclude that encryption provides full protection for data, both in terms of confidentiality and integrity. However, the above example shows that encryption used alone can be absolutely ineffective against some threats. If it is required to make "digital safes", then in addition to confidentiality, it is essential to at least assure data integrity. This can be done using two separate but combined mechanisms, or a mechanism designed to simultaneously guarantee confidentiality and integrity. However, such a mechanism has to be implemented very carefully.

41 **Synchronous** stream cipher is used when the cipher sequence is computed from the secret key regardless of the message to be encrypted. Conversely, **asynchronous** stream cipher or **self-synchronizing** stream cipher algorithms use encryption sequences that depend on the secret key and also on some bits of the ciphertext. The advantage put forward for this approach is that it enables a form of automatic resynchronization of the decryption, even if portions of the ciphertext are lost during the transmission.

[7] More precisely, this approach leads to a special class of stream cipher algorithms called "*binary additive stream cipher*" algorithms.

42 Note that such a property is based more on correction of transmission errors than on any cryptographic data protection. The question can then arise about the suitability of such techniques, and the actual threat that is to be avoided. Furthermore, some block cipher modes of operation such as the CBC method described above, also have the self-synchronization property provided that entire blocks are lost (see section 32).

F.1.1.3. Encryption security

43 Beyond the description of encryption primitives, a precise definition should be given of what is expected from their security properties. The intuitive concept according to which an attacker cannot deduce any information about the plain text message from the encrypted text is difficult to define precisely and should be done with a great deal of care. However, modern cryptography has developed very precise **security models** in order to clearly describe what is expected from encryption primitives.

44 Consider the randomization example described in paragraph 29, to clarify the difficulty of giving a precise definition of what is expected from encryption. One property that is naturally expected from an encryption algorithm is that it must be non-invertible by anyone without a secret key. Since there is no known attack, this property appears to be satisfied by the AES and the triple-DES. However, we have already seen that, no matter how good, a deterministic encryption will mean that two encryptions of the same message generate identical encrypted text, and that part of the information is thus exposed. Therefore, the non-invertibility property is insufficient and needs to be refined.

45 The idea according to which knowledge of ciphertexts must not reveal any information about the associated plaintexts is conventionally formalized by the notion of "semantic security". Such a definition is complex, but it can be summarized by the following experiment. Let us consider an encryption mechanism and chose two different messages[8] denoted M_0 and M_1. Let C be the ciphertext corresponding to one of the two messages. It should be impossible to determine which of the two messages was encrypted. This means that the view of a ciphertext does not reveal any information on the associated plaintext, regardless of the messages processed.

46 It is also possible to further reinforce this model taking account of the possibly very advanced capabilities of an attacker. For example, the previous experiment could be extended, allowing the person who proposes the two messages M_0 and M_1 to also obtain the encryption of any other message of his choice and the decryption of any other ciphertext, obviously different from C. If it is still impossible to guess if C is the encryption of M_0 *or* M_1, it is obvious that the encryption mechanism will be resistant to a large number of attack scenarios.

47 The strongest security models are obtained by a combination of a demanding security objective (for example semantic security) and a potentially very powerful attacker. Therefore, these models give the best guarantees. The highest security level for encryption mechanisms corresponds to the "indistinguishability against adaptive chosen ciphertext attacks " and is denoted by the acronym IND-CCA2.

[8] different but of the same size

48 The security models used in modern cryptography may appear exaggerated but they are justified by the fact that under some circumstances, attackers against whom it is required to provide protection have large capabilities in their attacks. Evaluating the security of a primitive in such security models can also minimize risks of a security fault when they are combined with other mechanisms so as to define complete systems. Furthermore, using reliable primitives faced with very powerful attackers is an important quality guarantee. Finally, modern cryptography proposes primitives capable of reaching such security levels without significantly reducing performances; consequently, it would be a shame not to use these mechanisms.

F.1.2. Authentication and integrity of messages

49 Some symmetric cryptographic techniques can also guarantee the integrity of transmitted data, regardless of whether they are protected in confidentiality or not. Such mechanisms are designed to guarantee that no data corruption has taken place during transmission. We have already mentioned that encryption mechanisms are unsuitable to guarantee such data integrity. Note also that cryptographic methods are generally aimed at guarding against deliberate and potentially intelligent attacks, unlike coding techniques and transmission protocols designed to detect or correct random and involuntary errors.

50 More precisely, the main technique by which data integrity is assured consists in calculating a **message authentication code** (also called MAC) by using the data to be protected and a secret key shared with the person to whom the message is intended. This authentication code, typically 128 bits long, is then added to the message and transmitted. After reception, the authentication code is recomputed using the secret key and the transmitted and potentially corrupted message. The result obtained is compared with the transmitted MAC; if they are identical, it is extremely probable that the data integrity has been maintained.

51 We emphasize the fundamental conceptual difference that exists between encryption and computation of the message authentication code. On the other hand, more technically, there may be a large number of similarities. The best known MAC algorithm is the CBC-MAC. The authentication code is then simply calculated by applying the CBC encryption method described in paragraph 30 to the message, without using any initialization vector (*IV*) and only using the last ciphertext block, over-encrypted with a key *K'*, different from that used in the CBC chain, as the MAC. Graphically, the schematic representation shown in figure 3 is obtained. It is clear that any modification (even minimal) of the message to be protected will lead to a completely different result for the MAC. However, it should not be thought that such a mechanism is reliable in the general sense based on this simple initial impression, because this is far from being the case.

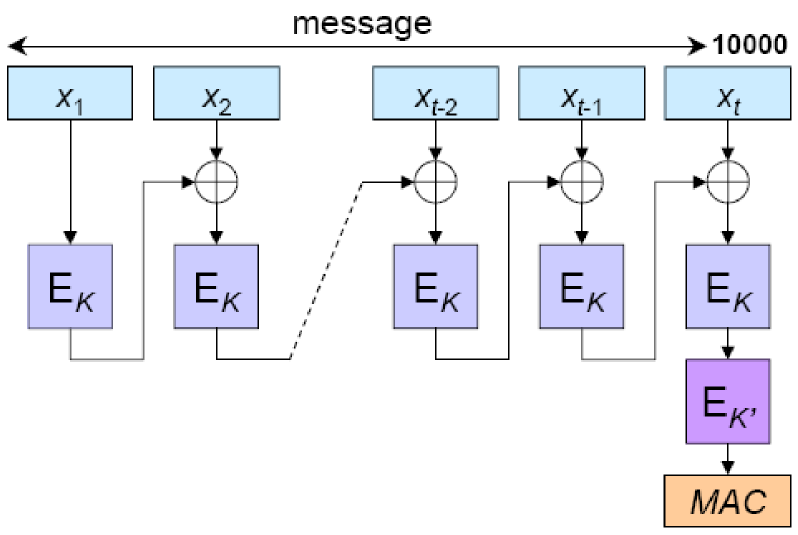


Figure 3. CBC MAC

52 Protection of the integrity of a message also guarantees its authenticity to a certain extent because knowledge of the secret key is necessary to generate correct MACs. On the other hand, this authentication has no value towards a third party because it does not make it possible to know the secret key holder who actually generated a MAC. Furthermore, some verification by a third party is only possible if the third party has been notified of the secret key used. This is intrinsically related to the symmetric nature of the mechanism.

53 Message authentication codes are sometimes denoted by the term **symmetric signature**, but obviously it should be understood that this signature qualification is unsuitable for the simple reason that the non-repudiation property is not assured. Only asymmetric mechanisms, that avoid the need to know the secret key to assess the validity of the signature, can genuinely render such security services.

54 Finally, note that a valid message authentication code guarantees authenticity of this message. If it is required to assure integrity and authenticity of a set of messages forming a communication, and for example to prevent replay or deletion of some messages accompanied by valid MACs, then special care shall be taken with the chaining methods used.

F.1.3. Authentication of entities

55 It is easy to derive mechanisms for authenticating entities, in the broad sense of the term, starting from the encryption and MAC computation primitives that we have just described,. The main difference between authentication of entities and authentication of messages is the **interactive** nature of authentication of entities, while authentication of messages must be possible in a single transmission, for example in secure messaging applications.

56 The most frequently used symmetric method to assure a correspondent's identity is to share a secret key with him. In order to assure that a person presenting an identity is actually who he says he is, the authentication will consist in making sure that this person

actually has the secret key. The simplest technique, derived directly from password techniques, consists in transmitting the secret. However, there are many disadvantages, notably the fact that a simple passive attacker can intercept the secret and then use it. Consequently, a much better method is the "*challenge - response*" method. The idea is to send an arbitrary message to the person to be authenticated, asking him to encrypt it using the shared secret key, and then send the result obtained. If the received response is correctly decrypted into the original question, there is a strong probability that the person who computed the encrypted message actually possesses the secret key and consequently is the person that he claims to be. However, note that this is only true if a challenge is never reused during subsequent authentications.

57 The problem of ensuring fresh challenges is very sensitive but it can easily be solved if a source of random data is available, without memorizing previously asked questions. It is possible to calculate the probability of the same *n*-bit question being obtained twice, if questions are selected at random. By application of the famous "birthday paradox", we see that the first collision occurs after the order of $2^{n/2}$ questions on average. Consequently, if the maximum number of authentications with the same key is significantly less than $2^{n/2}$, the use of random *n*-bit challenges is sufficient, since the risk of asking the same question twice is negligible. For application, 64-bit questions are perfectly suitable for several million authentications and 128-bit challenges are potentially usable with no limit that can be achieved in practice.

58 As is the case for encryption and integrity of messages, the major problem with such symmetric authentications lies in the need to share a secret key between the identifying person and the identified person. This also implies that it is impossible to distinguish between these two persons. Thus, the question arises whether it is genuinely necessary for the person verifying the identity to know the secret associated with the identity of an individual. Once again, the answer to this question is given by asymmetric cryptography, the basic principles of which are described below.

F.2. Asymmetric cryptography

59 The principal concept of asymmetric cryptography is that some operations are easy to carry out but are difficult to invert. A simple illustration is that it is easy, starting from a pot of yellow paint and a pot of blue paint, to simply mix them together to obtain a pot of green paint. However, the inverse operation of separating the yellow pigments from the blue pigments is impossible in practice, although it may be theoretically possible. Such invertible operations that are asymmetric in their difficulty do exist in the mathematical world. They are called difficult or asymmetric problems.

60 The best known of these asymmetric operations is multiplication of prime numbers i.e. numbers divisible only by 1 and by themselves. The inverse operation of finding these numbers starting from the result of the multiplication is called factorization into product of prime numbers. Choosing two fixed size prime numbers and multiplying them is an easy operation, but as soon as the size of the manipulated integers is sufficiently large, at the moment we do not know any efficient method of finding the prime factors. We should emphasize the fact that such factorization is not impossible; it is perfectly defined mathematically and very simple factorization algorithms are known. The problem is the complexity in time, i.e. in the computation time necessary for these methods to find the result. Furthermore, there is no way of knowing that an efficient method does not exist. All that can be said at the moment is that no efficient factorization method using an available technology has been published for now.

61 "Trapdoor" asymmetric problems are a special kind of asymmetric problems. They are based on an operation that is easy to carry out but is difficult to invert unless one knows an additional element called a "trapdoor". The best known example, but also one of the few examples that can be used in practice, is the operation on which the famous RSA cryptosystem is based. This operation is directly related to the factorization problem, knowledge of the prime factors forming the trapdoor allowing to easily invert the operation.

62 As an illustration, we might think about mixing iron and copper filings. As in the case of paint pots with different primary colors, it is easy to perform the mixing operation but the inverse operation is very complex, unless there is a magnet that acts as a sort of trapdoor.

63 Cryptographic research during the past 25 years has provided a number of asymmetric mathematical problems that can be used for cryptographic purposes. However, they are very few and almost all of them are based on problems derived from number theory; factorization and computation of "discrete logarithms" in a variety of mathematical structures including "elliptic curves", are particularly interesting examples.

64 The basic idea of discrete logarithm problems consists in using a mathematical structure containing a finite number of elements that can be combined using an operation with properties similar to addition or multiplication on classical integers. Such a structure is called a "group" in algebra. An iterated version of the operation acting on elements in the group can then be defined. If the operation is noted multiplicatively and if x denotes an element of the group, then the product of n copies of x is denoted "x to the power n" namely x^n. In general, it is easy to calculate x^n, even for very large values of the integer n. However, in some groups, the inverse operation of starting from x and x^n and then finding the integer n is very difficult, although it is perfectly defined mathematically. This operation is called "discrete logarithm", where n is called the "logarithm of x^n in base x". As in the case of factorization, it must be clear that the logarithm computation is not impossible in theory; for example it suffices to test all possible values of n, although there are much better methods. However, for sufficiently large groups, we currently do not know any efficient method of performing such computations in a reasonable time for the kind of groups used in cryptography. "Elliptic curves" with carefully chosen parameters are such examples of groups, in which the computation of discrete logarithms is believed to be particularly difficult.

65 In this description, we will not go into details requiring any complex mathematical reminders. Simply note that these problems only become genuinely difficult, and therefore cryptographically interesting, when some of the parameters are sufficiently large. These sizes determine the security intrinsically provided by the basic problem to the whole cryptographic protocol. Therefore the security of an asymmetric system is evaluated as a function of the difficulty of numerically solving a given mathematical problem and not as a function of the size of the key space. Thus for example, references to RSA-1024 mean that RSA is used with a 1024-bit long parameter called the modulus, obtained by multiplication of two prime numbers of 512 bits each. Consequently, it should not be surprising that 1024-bit or larger parameters or keys are used in asymmetric cryptography, while keys in symmetric cryptography rarely exceed 128 bits or even 256 bits for the longest keys. Therefore the sizes of symmetric and asymmetric keys cannot be compared.

66 Finally note that the use of a genuinely difficult and correctly sized problem is a necessary condition to obtain a required security level. Obviously, this is far from being sufficient because there are many other security weaknesses in using a given problem, in modes of operation, in management of keys, in problems of design or implementation of protocols, etc.

F.2.1. Asymmetric encryption

67 The essential concept of **asymmetric encryption**, frequently called public key encryption, is that nothing requires the sender of an encrypted message to be capable of decrypting the messages that he sends. In other words, although it appears quite natural, the concept by which the same key must be used both for encryption and decryption is not fundamentally justified.

68 In the framework of encryption, the concept is to use **key pairs** composed of a **private key**[9] and an associated **public key**. In order to encrypt a message for the attention of a correspondent, the correspondent's public key will be used. After transmission, the inverse decryption operation is carried out using the private key. Properties of the mechanism are such that knowledge of the public key is not sufficient to determine the private key. Consequently, the public key, as its name suggests, can be widely distributed. On the other hand, the private key must be kept confidential.

69 A classical illustration consists of imagining a safe, as they are frequently used, requiring a secret combination to be opened but that anyone can close. The equivalent of the public key is the safe, freely accessible and available to any sender. The private key is the combination used to open the safe. In order to transmit a document, it is "sufficient" to recover an open safe belonging to the addressee, put the document in question into it and send it. On reception, knowledge of the private key opens the safe and releases the document. Obviously, this image quickly reaches its limits, particularly in terms of availability of safes. On the other hand, distribution of public keys is much easier in the digital context. Furthermore, a single public key can be used by many senders, while as many copies of safes as necessary must be produced. Finally, remember that no integrity is necessarily guaranteed, as is the case for symmetric encryption.

70 It is important to notice that unlike symmetric mechanisms, in this case there is no need for two correspondents to share a secret key in advance. Theoretically, this means that problems of coming to an agreement about keys inherent to symmetric cryptography are very elegantly solved. However, the problem of certification of public keys arises to assure that a public key used for encryption actually belongs to the person to whom the message is intended.

71 As when using a block cipher, the use of public key encryption with arbitrary sized messages must be very precisely specified. This means that it is necessary to format and pad messages in order to apply the encryption algorithm. The use of random data is also necessary to "randomize" encryption and consequently to prevent encryption of the same message twice from producing identical ciphertexts. This is of fundamental importance for applications in which the message space, i.e. the set containing all messages for this application, is restricted to a small subset of possible messages. This problem is also similar to the problem encountered in the symmetric case and described starting from paragraph 27.

[9] Current practice requires that the term “secret key” should be reserved for symmetric applications and the term “private key” should be reserved for asymmetric applications. Therefore, a public key is usually naturally associated with a private key, whereas such an association is meaningless for a secret key.

72 Finally, note that for reasons of efficiency, there is no point in encrypting large messages using an asymmetric mechanism. A much more efficient method, referred to as hybrid encryption, consists in choosing a session key for a symmetric encryption mechanism and only transmitting this session key, encrypted with the public key mechanism. Thus, the session key can be transmitted to the contact in a secure manner. Any message can then be conventionally and very efficiently encrypted with the session key.

F.2.2. Digital signature

73 A **digital signature** guarantees integrity of a message without using a secret key shared between the sender and the receiver. It also provides strong authentication of the message sender, by preventing the message sender from subsequently denying having sent the message; this is the non-repudiation property.

74 Signature is an asymmetric mechanism that has at first sight many similarities with public key encryption. However, signature and encryption should not be confused. The main common feature is the use of key pairs consisting of a private key and an associated public key. The private key is used to generate the signature of a message. The signature is then appended to the message[10], like MACs described in chapter F.1.2. All that is necessary to assess the validity of a signature is to know the public key. Consequently, anyone can potentially check whether a message is authentic because the public key can be made freely available. Therefore, this is very much like an electronic version of a conventional handwritten signature, and some would even say it is better than the traditional signature. However, note once again that the certification of the public key is a crucial problem that we will mention quickly below under chapter F.3.3 dedicated to key management, and that is repeated in more detail in the reference document entitled "Management of cryptographic keys – Rules and recommendations related to management of keys used in cryptographic mechanisms"[11], dedicated specifically to this problem.

75 As for encryption, the formatting to be applied to the message before signature is very important in terms of security. It frequently uses a hash function transforming an arbitrary length message into a small fixed size hash.

76 It is often tempting to present digital signature as being a sort of reciprocal for asymmetric encryption. This is due to the fact that in the case of RSA, encryption and signature are actually very similar, since the signature of a message is very much like the decryption operation. However, this case is exceptional, and most signature schemes cannot be used for encryption purposes.

F.2.3. Asymmetric authentication of entities and key exchanges

77 Concepts for interactive authentication of entities described in chapter F.1.3 can obviously be extended to asymmetric primitives. For example, all that is necessary to authenticate someone whose public key is known with certainty is to encrypt a sufficiently long message for his attention and ask him to return this message in plain text. Similarly, we can ask him to sign such a message and then verify that this signature is valid.

[10] More precisely, the mechanism of computing a signature and adding it to the message is called "signature with *appendix*". Other techniques may be considered so as to save a little space with mechanisms such as RSA based on trapdoor permutations. However, this is not very important in a first approach.

[11] Currently available only in French

78 However, there are mechanisms specially designed for the interactive framework. These primitives called "zero knowledge" primitives are the source of most digital signature schemes, although they have not been used much in practice. For example, the American DSA (*Digital Signature Algorithm*) standard is derived directly from the Schnorr authentication protocol.

79 Asymmetric methods are also helpful for key agreement, that is, to enable two correspondents to establish an initial, common secret key although the network is potentially controlled by an attacker. The initial work of W. Diffie and M. Hellman [1] addresses this problem only (not asymmetric encryption).

80 Technically, the basic comment by W. Diffie and M. Hellman is that, provided a mathematical structure in which computing discrete logarithms is hard (see section 64), then choosing a large secret integer a and publishing x^a should not reveal any information about a. Thus, for two contacts A and B to reach a common secret K, it suffices that A chooses an integer a and sends $y = x^a$, and that B chooses an integer b and transmits $z = x^b$. A then computes $K = z^a = (x^b)^a = x^{(ab)}$ and B computes the same value $K = y^b = (x^a)^b = x^{(ab)}$. On the other hand, an attacker who is passively listening to the communication can only learn y and z. Indeed, at the moment, for an attacker to recover the shared secret K, we do not know any more efficient method than computing first a discrete logarithm to find a or b.

81 However, note that this elementary protocol cannot guarantee security faced with active attackers, capable of modifying the communications. Nor does it provide any form of authentication of the contacts.

82 In practice, authentication and key agreement mechanisms are generally used together because firstly, there is not much point in sharing a key with a person whose identity is not known, and secondly, authentication is not very useful on its own. However, the tie between authentication and key agreement must be made carefully.

F.2.4. Security of asymmetric primitives

83 Modern cryptography has developed mathematical techniques in order to attempt to prove the security of primitives, particularly asymmetric primitives. These proofs are not absolute in nature, but above all are based on a security model that formally defines the expected properties of the primitives, and the assumed capabilities of the attacker against whom protection is required.

84 These proofs are said to be "by reduction" in the sense that they will reduce the global security of a primitive to a clearly identified assumption such as "it is computationally infeasible to factorize 2048-bit RSA moduli". Therefore, there are practically never any proofs in the sense of information theory, i.e. assuring security faced with an attacker with infinite power (see section 37).

85 In order to illustrate the importance and also the difficulty in defining a security model, consider the example of the signature. We can firstly consider various types of attacks depending on the capabilities of the attacker, which can vary from simple knowledge of the public signature verification key to the capability of obtaining the signature of any message of his choice. It would also be relevant to consider the definition of what is meant by a successful attack. This can vary from a simple "existential forgery" consisting in successfully generating a valid signature of an uncontrolled message, up to a "total break" which consists in finding the private signature key.

86 Obviously, an asymmetric primitive proven to be secure based on clearly identified and reasonable assumptions, is all the more attractive when the proof considers powerful attackers and modest definitions of a successful attack. In the same

way as for the security of symmetric encryption described in chapter F.1.1.3, security models used in modern cryptography may appear excessive, but experience has shown that this is far from being the case and that it is important to consider this restricting framework in order to correctly evaluate primitives.

F.3. Other cryptographic primitives

87 Some cryptographic primitives cannot be classified in an approach opposing symmetric and asymmetric principles. This is the case particularly for primitives that do not directly use secret elements such as keys .

F.3.1. Hash functions

88 The most important of these primitives is hash functions. Its purpose is to deterministically transform an arbitrary long sequence of bits into a fixed size **digest**, called a **hash**. A cryptographic hash function is also required to be non-invertible in practice, in other words, given the digest, it is impossible to find a message for which the image through the hash function is equal to this digest. Another requirement is that it is impossible to find two distinct messages with the same digest. It is said that the hash function is **collision resistant**.

89 Obviously, since a hash function maps an infinite set to a finite size set, there is an infinite number of such collisions. However, the requirement is that it should be impossible to compute such a collision in practice in a reasonable time.

90 Once again, the birthday attack may be applicable; if *n* denotes the number of bits of the digest and if the hash function can be considered as having a relatively random but deterministic behavior, then the computation of the order of $2^{n/2}$ hashes of random messages is sufficient for two of these messages to lead to the same digest. This gives a lower bound to the size of outputs for cryptographic hash functions.

91 Hash functions are essential for digital signature schemes in order to reduce the arbitrary length message to a simple small fixed size digest. They are also used to create some message authentication codes.

F.3.2. Generation of cryptographic random numbers

92 Cryptography makes intensive use of random data, typically to generate private or secret keys but also for many other applications such as formatting messages before encryption or signature. The quality of the random data used may be critical for security and motivates the use of “high quality” random data.

93 Consider the American DSA signature standard as a particularly striking example of the need to have good random data. In using this algorithm, the signature of a message requires the use of a 160-bit random number, kept secret by the signatory. A new random number has to be provided for each signature, independently of the previous numbers and therefore for one use only.

94 Although this does not affect the practical security of DSA, an attack is known at the present time to recover private signature keys, provided that signatures are available for which only 3 bits of the single-use random number are known. This attack works very efficiently even if the remaining 157 bits are perfectly random and unknown to the attacker.

This example shows that for some applications, it is impossible to be satisfied with partially random numbers.

95 The generation of true random bits, i.e. equal to 0 or 1 with the same probability and, above all, independent of each other, is particularly difficult on a fundamentally deterministic platform such as a computer or, even worse, a smart card microprocessor. However, there are special physical devices that make use of supposedly random phenomena such as thermal noise or time between two disintegrations of a radioactive source. This is referred to as the "**true random number**" or "**physical random number**" generator.

96 A "**pseudo random number**" can also be generated, i.e. sequences of bits that cannot be distinguished from true random sequences, but that are derived from a deterministic mechanism initialized with a small "germ" or "seed" that itself is truely random. Most stream cipher mechanisms are built around such pseudo-random generators.

97 In order to illustrate the concept of a pseudo random number generator, we will quickly describe the generator standardized by ANSI as reference X9.31, also previously described in ANSI X9.17. It uses a secret key *K* and a block cipher primitive denoted E_K. This iterative process uses an internal variable, kept secret and denoted $Etat_i$. It is updated during each iteration by the mechanism described below in Figure 4. The initial value of the state variable and the key *K* form the germ. The block denoted $ALEA_i$ contains pseudo random data generated during the ith iteration. Finally, the block Ext_i may contain optional external data, for example to diversify the mechanism using poor quality random data, for example derived from the clock.

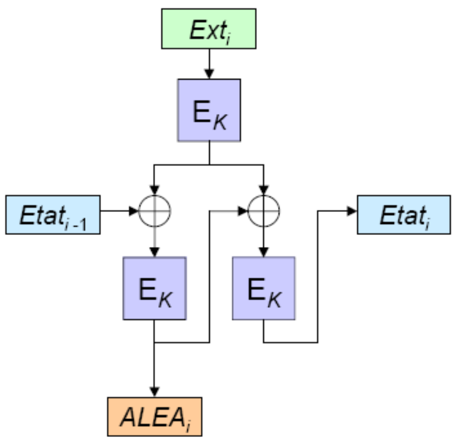


Figure 4. X9.31 pseudo random number generator

98 Finally, note that by definition in practice it is impossible to distinguish a correctly generated pseudo random number from true random bits. Despite the existence of clearly defined theoretical concepts such as entropy, derived from information theory, the only way of testing the generated bits is to apply statistical tests in order to detect statistical biases for which the probability of occurrence is negligible in sequences of bits derived from true random number sources. However, the efficiency of these tests, no matter how sophisticated they are, is very limited in practice.

F.3.3. Key management

F.3.3.1. Symmetric secret keys

99 Key management may be more or less simple depending on the applications. In the context of symmetric mechanisms, the main difficulty lies in the distribution, or key agreement so that correspondents can share the same initial secrets without them being intercepted by potential attackers. This can be done using modern asymmetric techniques, but it may also be done using non-cryptographic organizational methods.

100 A maximum life called the **crypto-period** is also usually associated with each key. Such a life can be defined by a usage deadline or by a counter of the number of uses that must not exceed a certain limit. Such a limitation to the use of a key is usually intended to mitigate the effect of keys being compromised. However, it may also be necessary if cryptographic primitives are undersized to achieve the required security level. However, it is important to understand clearly that in a cryptographically well-designed system, there must be no "wear" phenomenon of keys limiting their use.

101 In order to protect keys during storage, the keys may themselves be encrypted with another key that in general does not need to be shared. Usually, a key encrypted in this way is referred to as a **black key**, unlike **red keys** that are kept in plain text. It is obvious that the set of keys used in a system in operation cannot all be black at the same time.

102 Finally, note a special case of architecture, still used fairly frequently, that uses a secret shared widely between a large number of users. Disclosing such keys usually has dramatic consequences in terms of security, which is contradictory to their widespread usage. In some applications, the exclusive use of symmetric primitives makes such architectures necessary; this strongly argues in favor of the use of asymmetric architectures so that widely shared keys are not necessary.

103 For example, imagine a large group of *n* individuals who would like to mutually authenticate each other. With symmetric techniques, either a secret key can be provided for each pair of individuals, which implies that each person memorizes at least *n*-1 keys, or the same key can be given to everyone. If it is also required to be able to add new members easily, the second solution becomes the only possible solution. However, what confidence can be placed in such a system, even if the key is stored in a protected location such as a smart card?

104 One simple means of solving this problem with an asymmetric technique is to choose a key pair for each member of the group, for which the public key is certified by an authority. Therefore, each member needs to store only his own key pair and the public key of the authority. Any one of the identification techniques mentioned in chapter F.2.3 can then be used.

F.3.3.2. Asymmetric key pairs

105 Key pair management in asymmetric cryptography is both simpler and more complex than in the symmetric case. It is simpler and also more reliable because there is no longer any need to share secrets with several persons. Thus, a private key needs to be known only by its holder and certainly not disclosed to anyone else. Consequently, in theory there is no need to have a third party generate such keys. For example, it is quite conceivable that a private key is generated by a smart card and that this key will never leave the supposedly secure location of the card at any time during the life of the system.

106 However, the major problem that remains is the need to associate a public key with the identity of its legitimate holder. Certification of public keys can be achieved using signed certificates issued by an authority that consequently certifies that a given public key is actually assigned to a particular individual or entity. The problem then arises of verifying this signature which in turn requires knowledge of the public key of the authority. One way of certifying this key would be for a higher authority to generate a new certificate, and so on. This then leads to a path of trust leading to a root key, which must eventually be trusted. Such constructions are referred to as **public key infrastructures (PKI)**.

107 Finally, note that for many practical applications, a sort of emergency escrow is necessary, for example to access encrypted data without necessarily being the addressee of this information. There may be many motivations for such **recovery mechanisms** but it is important to emphasize the fact that they may be perfectly legitimate and legal. The simplest method is key escrow consisting in putting private or secret keys under seals while checking conditions of access to this information. However, modern cryptographic work offers many more flexible, reliable and efficient solutions.

G. Academic elements for cryptographic key sizes

This appendix contains some information derived from academic announcements and publications that contribute to justifying cryptographic key sizes.

G.1. Cryptographic computation records

Cryptographic computation records are used to estimate a lower limit to the practical difficulty associated with some problems. Essentially, they relate to the "breaking" symmetric key algorithms by enumerating all possible keys (so-called exhaustive search) and to solutions of mathematical problems derived from number theory (factorization and discrete logarithm in varied structures).

G.1.1. Computation records in symmetric cryptography

The main publicized computation records used the Internet network and volunteer Internet users who agree to carry out the computations on their personal computer as "background task". The main results concern the breaking of 56-bit DES keys:

- June 17 1997, **96 days** of distributed computation on the Internet,
- February 23 1998, **41 days** of distributed computation on the Internet[12],
- July 17 1998, **56 hours** of computation on a dedicated machine[13], at an estimated cost of $ 250 000,
- January 19 1999, **22 hours** of computation combining the machine mentioned above and computations on the Internet

and secondly breaking of RC5 encryption keys:

- October 19 1997, **56-bit** version broken after 250 days of computation on the Internet,
- July 14 2002, **64-bit** version broken after 1757 days of computation on the Internet.

Obviously, these "records" do not take account of the computing capacities of specialized government services which of course do not publish their work. However, this should not generate excessive concerns, as described in section F.1.

[12] See http://www.distributed.net
[13] See http://www.eff.org/Privacy/Crypto/Crypto_misc/DESCracker

G.1.2. Factorization records

The main successive computation records for factorization of moduli that are the products of two comparably sized prime numbers are:

- June 1993, **399** bits (120 digits)
- April 1994, **429** bits (129 digits)
- April 10 1996, **432** bits (130 digits)
- February 2 1999, **466** bits (140 digits)
- August 22 1999, **512** bits (155 digits)
- January 2002, **524** bits (158 digits)
- April 1 2003, **530** bits (160 digits)
- December 3 2003, **576** bits (176 digits)
- June 9 2005, **663** bits (200 digits)

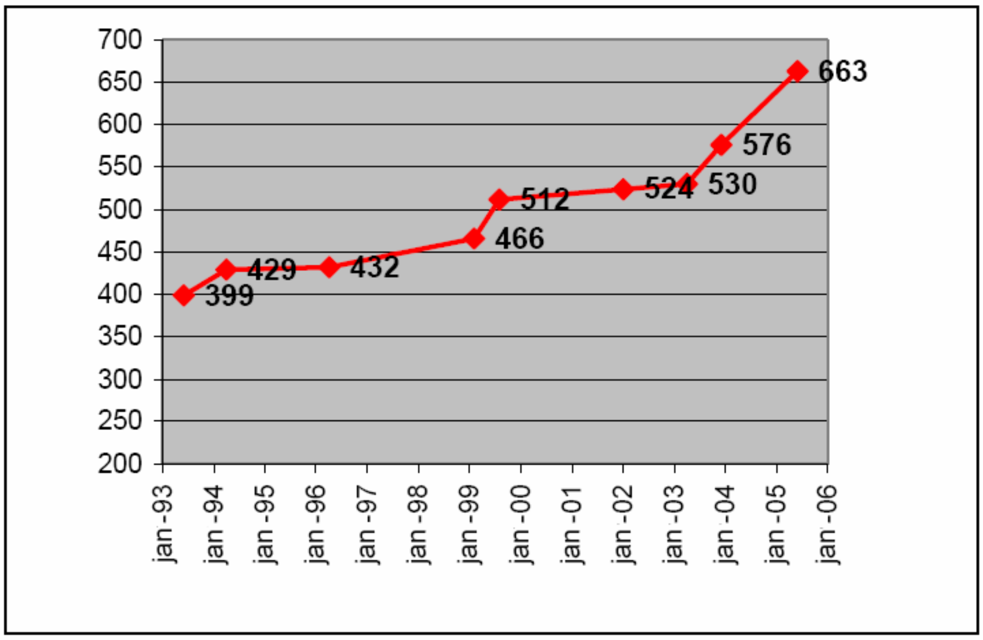


Figure 5. Factorization records

The first two records used the quadratic sieve algorithm and the others used the number field sieve algorithm, which is the most efficient known at the moment. Most challenges were put forward by the RSA Company[14].

[14] See http://www.rsasecurity.com/rsalabs/challenges

Factorization by dedicated machines

In the same way as dedicated machines can be imagined and designed exclusively to carry out exhaustive search on symmetric encryption keys, it is now seriously envisaged to design such machines to factorize large RSA moduli. The most successful project was presented by Adi Shamir and Eran Tromer in August 2003. Cost estimates indicated that it would apparently be possible to make a machine capable of factorizing 1024-bit moduli in less than a year, for a price of a few tens of millions of Euros. At the moment, no specific design announcement has been made.

Other factorization records

Finally, one should note that in some cases, special factorization algorithms that are more efficient than general algorithms can be used, but they are not applicable to all integers (and particularly to RSA moduli so far).

The SNFS algorithm (so-called "special" algebraic sieve) has thus been used to factorize an integer with 244 digits (i.e. 809 bits) at the beginning of 2003, which is an integer much larger than the last RSA record so far.

G.1.3. Computation records for discrete logarithm in GF(p)

The main successful records in terms of discrete logarithm computation in a prime Galois field with *p* elements GF(p) are:

- September 25 1996, Weber, Denny and Zayer, **281** bits (85 digits)
- May 26 1998, Joux and Lercier, **298** bits (90 digits)
- 2000, Joux and Lercier, **331** bits (100 digits)
- January 19 2001, Joux and Lercier, **364** bits (110 digits)
- April 17 2001, Joux and Lercier, **397** bits (120 digits)
- June 18 2005, Joux and Lercier, **432** bits (130 digits)

in which the bit size denotes the size of the prime number *p*.

G.1.4. Computation records for discrete logarithm in GF(2^n)

The main successful records in terms of discrete logarithm computation in a finite Galois field with 2^n elements GF(2^n) are:

- 1992, Gordon and McCurley, GF(2^{401}) namely **401** bits
- September 25 2001, Joux and Lercier, GF(2^{521}) namely **521** bits
- February 23 2002, Thomé, GF(2^{607}) namely **607** bits
- September 22 2005, Joux and Lercier, GF(2^{613}) namely **613** bits

Note in particular that the discrete logarithm computation is much easier in GF(2^n) than in GF(p), for equivalent field sizes.

G.1.5. Computation records for discrete logarithm in $GF(p^n)$

The main successful computation records for a discrete logarithm in a Galois field with p^n elements $GF(p^n)$ for p greater than 2, are:

- June 28 2005, Lercier and Vercauteren, $GF(370801^{18})$ namely **336** bits (101 digits)
- October 24 2005, Joux and Lercier, $GF(65537^{25})$ namely **400** bits (120 digits)
- November 9 2005, Joux and Lercier, $GF(370801^{30})$ namely **556** bits (168 digits)
- August 2006, Joux and Vercauteren, $GF(p^3)$ namely **400** bits (120 digits)

Note in particular that for equivalent field sizes, the discrete logarithm computation for p and n of medium size is intermediate in difficulty between the difficulty in $GF(2^n)$ and GF(p).

G.1.6. Computation records for discrete logarithm on elliptic curve

Challenge	Announcement date	Number of operations
ECCp-79	06/12/1997	$\mathbf{2^{40}}$
ECCp-89	12/01/1998	$\mathbf{2^{44}}$
ECCp-97	18/03/1998	$\mathbf{2^{47}}$
ECCp-109	06/11/2002	$\mathbf{2^{54}}$
ECCp-131	*Unsolved*	
ECC2-79	16/12/1997	$\mathbf{2^{40}}$
ECC2-89	09/02/1998	$\mathbf{2^{44}}$
ECC2-97	22/09/1999	$\mathbf{2^{47}}$
ECC2-109	15/04/2004	$\mathbf{2^{54}}$
ECC2-131	*Unsolved*	
ECC2K-95	21/05/1998	$\mathbf{2^{44}}$
ECC2K-108	04/04/2000	$\mathbf{2^{51}}$
ECC2K-130	*Unsolved*	

On November 6 1997, the Certicom Company published a list of challenges[15] on the problem of the discrete logarithm on an elliptic curve. There are three types of challenges: firstly "arbitrary" elliptic curves defined on GF(p), secondly "arbitrary" elliptic curves defined on $GF(2^n)$, and thirdly Koblitz curves also defined on $GF(2^n)$. These challenges are denoted by the codes ECCp-x, ECC2-x and ECC2K-x respectively, where x is the bit size of the prime order of the subgroup in which the operations are defined.

[15] See http://www.certicom.com

The above table shows the results announced so far, upcoming unsolved challenges as well as the number of operations that were necessary.

G.2. Study of key sizes from the article by Lenstra & Verheul [2]

In 2001 Arjen Lenstra and Eric Verheul published an article which aimed at comparing robustness levels of different problems used in cryptography. Obviously, this academic work must be considered with a great deal of caution; however, it does have the merit of suggesting motivated and reasonable models.

Warning: this work is mentioned in the document because it is the most complete document produced in this field by academic research workers so far. It is often referenced. However, the fact that it is quoted and that some of the results are produced in this appendix is not intended to give any credit to the article as a whole. In particular, authors are responsible for some of their declarations.

G.2.1. Evolution of symmetric key sizes

It is easy to estimate the evolution of the necessary sizes of symmetric keys based on Moore's law, according to which the memory quantity and the available computation power for a fixed price double every 18 months. The following graph indicates the increase in key sizes necessary to maintain the security level of a DES (56 bits) in 1982. It is read as follows. The year is given as the abscissa. The ordinate indicates the key size in bits to provide security equivalent to the DES security level in 1982.

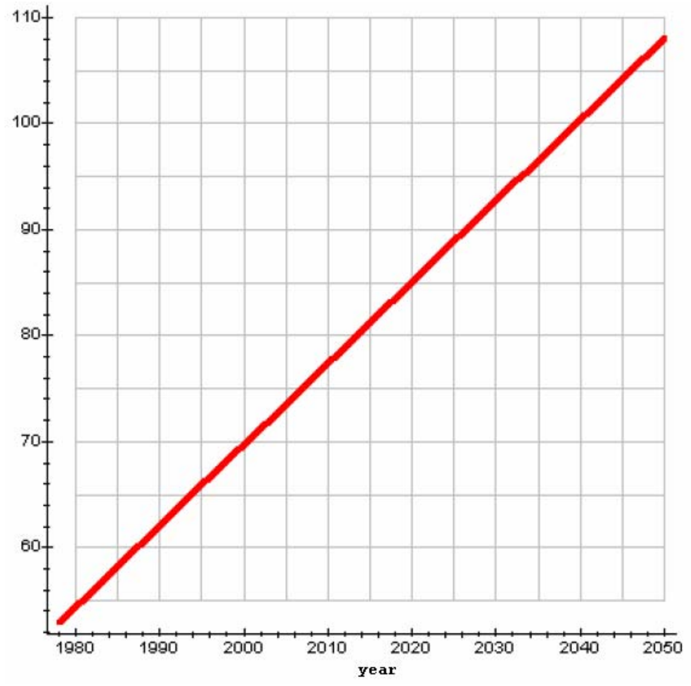


Figure 6. Symmetric key size offering security equivalent to the DES security level in 1982

Comments:

- 128-bit keys provide a very high security level against an exhaustive search attack,
- 256-bit keys provide an even stronger security level against exhaustive search attacks,
- In conclusion, efficient protection can be provided against exhaustive search attacks, even in the very long term.

G.2.2. Evolution of moduli sizes in asymmetric cryptography

The same type of evaluation can be made for asymmetric moduli. No distinction is made herein between the factorization problem and the discrete logarithm problem in GF(p) because this distinction would not be very meaningful in practice. However, it is considered that the discrete logarithm problem is slightly more difficult than the factorization problem; in practice, it can be considered that there is a difference of 100 to 200 bits between them in the size ranges in which we are interested, this approximation being very imprecise. The following graph is read as follows: the year is given as the abscissa, and the ordinate indicates the modulus size in bits to provide security equivalent to the DES security in 1982.

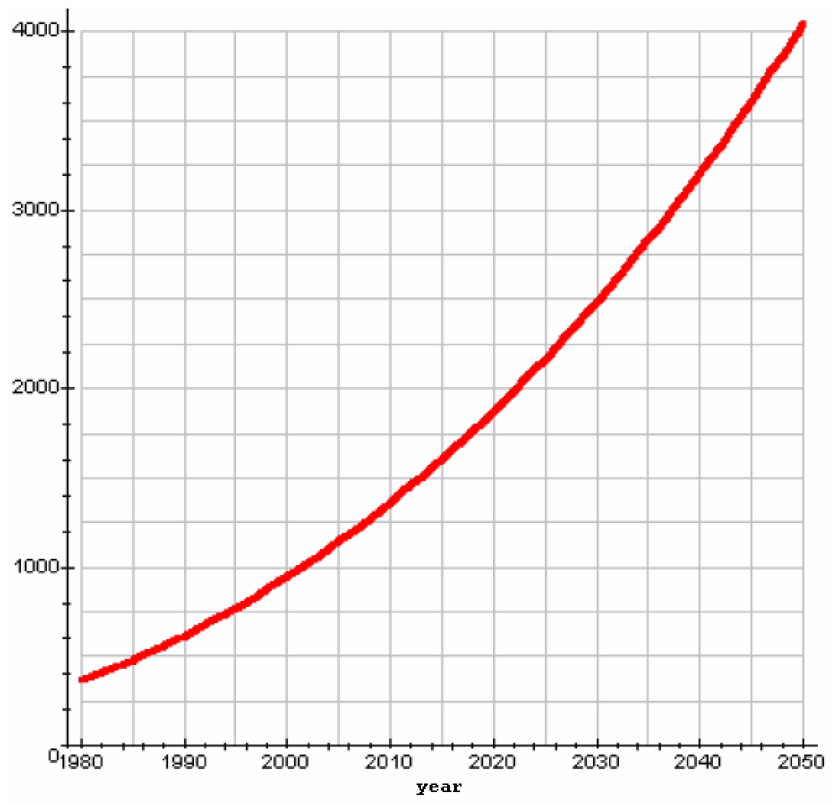


Figure 7. Modulus size providing a security level equivalent to the DES security level in 1982

Comments:

- Growth is significantly non-linear due to the existence of so-called "sub-exponential" algorithms such as quadratic and number field sieves.
- The use of large moduli will be necessary in the medium term.

A comparison should be made between this curve and publicly announced records. Obviously, there is no risk of observing an agreement between these two approaches because the curve in Figure 7 indicates minimum modulus sizes to be used to provide cryptographic security, while records only translate academic know-how. There is necessarily a difference between these two approaches, which some consider to be very important.

For comparison, remember that in the case of symmetric keys, the idea of using 65-bit keys would not be considered because a public effort has been made on Internet to find a 64-bit key. Similarly, using 674-bit moduli based on the fact that a 664-bit modulus has been factorized, where 674 bits is obtained by increasing 664 bits by one 64th, would appear to be a risky approach.

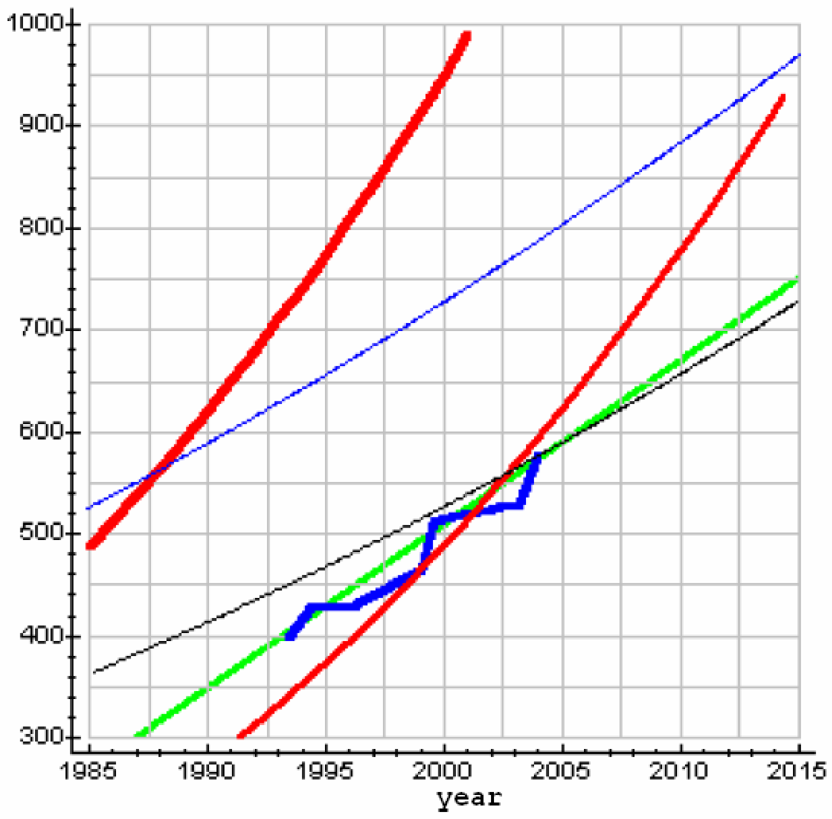


Figure 8. Comparison between factorization records and theoretical sizes necessary for the security level equivalent to the DES security level in 1982.

These precautions having been taken, consider published records and compare them with previous curves; in the above figure,

- the thick blue curve corresponds to published factorization records,
- the thick red curve at the top left corresponds to the theoretical curve in Figure 7
- the thin red curve corresponds to this same curve offset horizontally by 15 years,
- the green curve is an attempt at linear interpolation between points in the blue curve,
- the thin black curve is obtained by extrapolation of the most recent factorization record using the theoretical complexity of the algorithm used (the algebraic sieve),

- ♦ the thin blue curve corresponds to the same curve as above but assuming that there is 1,000 times more computation power available.

What can we conclude from this graph? Firstly, it is obvious that there are few experimental points, and therefore that they are not very significant. Furthermore, the computation effort necessary to obtain each of these records was not considered. However the theoretical curve offset by 15 years can be considered as an interpolation of records. In this case, we can say that in general, following the minimum modulus sizes recommended by Lenstra-Verheul would provide protection in practice against public attacks for fifteen years beyond the planned usage end date.

On the other hand, graphically (i.e. ignoring all other theoretical considerations) it would appear that records can equally well be interpolated by a straight line….

In conclusion, elementary cryptographic principles suggest that all available assumptions should be taken into account to size the system correctly to provide protection against the most powerful attackers. This is the aim of Lenstra & Verheul, and also of most experts in the field. However, the observations that we have just made could encourage some designers to take the risk of using smaller moduli so as to gain performance while maintaining an "apparently sufficient" security. Obviously, the debate could go on forever because it is impossible to make an objective decision using unquestionable arguments.

G.2.3. Evolution of elliptic curve sizes

The same work is done for elliptic curves. The two projected curves given below are obtained assuming that hypothetical algorithmic progress will be made (red curve) or will not be made (blue curve). The graph is read as follows: the year is given in the abscissa, while the ordinate indicates the modulus size in bits necessary to provide a security level equivalent to the DES security level in 1982.

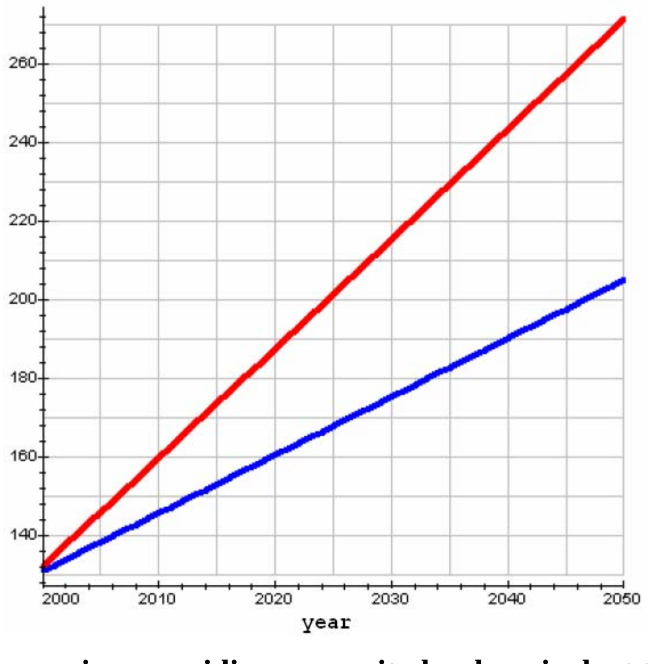


Figure 9. Elliptic curve sizes providing a security level equivalent to the DES security level in 1982.

G.2.4. Equivalent security between asymmetric modulus size and symmetric key size

The above data can be used and equivalences can be plotted regardless of time, so as to compare symmetric and asymmetric keys. The following curve is obtained by comparing symmetric keys and asymmetric moduli. The graph is read as follows: the number of bits of symmetric keys is given in the abscissa, while the ordinate indicates the bit size of the asymmetric modulus providing an equivalent security level.

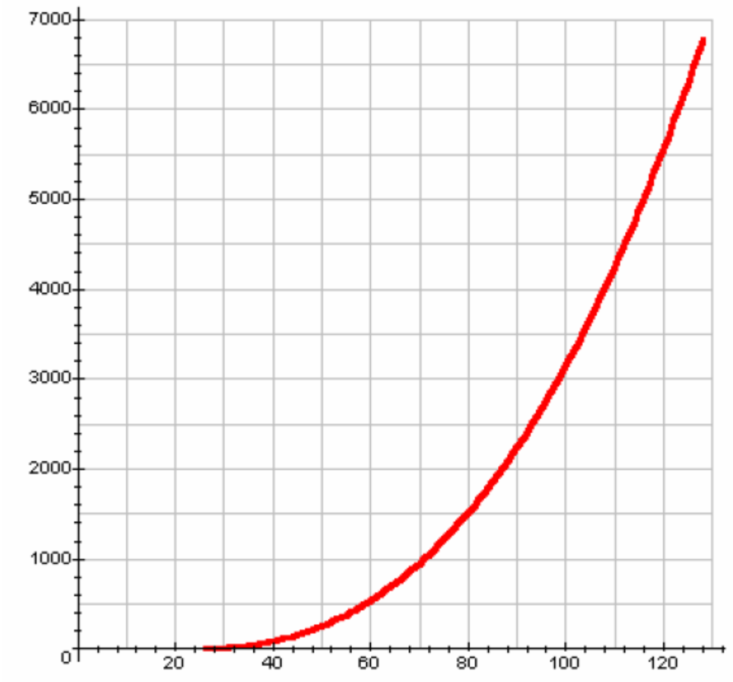


Figure 9. Equivalence between the asymmetric modulus size and symmetric key size

<u>Comments:</u>

- Searching for security in asymmetric cryptography based on the factorization problem or on the discrete logarithm problem equivalent to the security provided by symmetric cryptography on a secret key with more than 128 bits results in a very large key size, therefore not easily usable in practice.
- It is absurd to attempt to use moduli with a security comparable to the security of a 256-bit symmetric key.
- It is frequently attempted to compare the size of symmetric keys and the size of asymmetric keys by estimating the equivalent "*number of asymmetric bits*" equivalent in terms of security to a "symmetric bit". In other words, this is simply the slope of the curve in Figure 9. Consequently, in common terms we can state that:
  - "for keys about 512 bits long, a symmetric bit is equivalent to 34 asymmetric bits”,
  - "for keys about 1024 bits long, a symmetric bit is equivalent to 51 asymmetric bits”,
  - "for keys about 2048 bits long, a symmetric bit is equivalent to 77 asymmetric bits”.

◆ These statements can be rephrased as follows: *"to double the effort necessary to factorize a 1024-bit modulus, a 1024 + 51 = 1075 bit modulus should be used".*

G.2.5. Equivalent security between elliptic curve size and symmetric key size

The same process can be applied by comparing symmetric keys and cryptosystems based on elliptic curves. The two projected curves given below are obtained assuming that hypothetical algorithmic progress will be made (red curve) or will not be made (blue curve). The graph is read as follows. The bit size of a symmetric key is given in the abscissa. The ordinate indicates the bit size of the elliptic curve necessary to provide equivalent security.

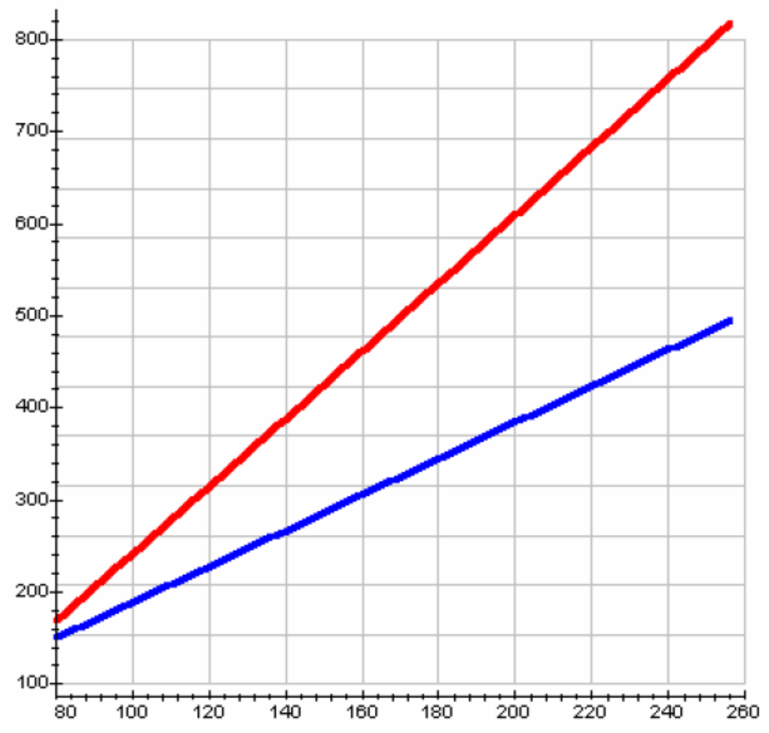


Figure 10. Equivalence between cryptosystem size based on an elliptic curve and symmetric key size

End of document