BS ISO/IEC 9796-2:2010

BSI Standards Publication

Information technology — Security techniques — Digital signature schemes giving message recovery

Part 2: Integer factorization based mechanisms

bsi.

...making excellence a habit.™

National foreword

This British Standard is the UK implementation of ISO/IEC 9796-2:2010. It supersedes BS ISO/IEC 9796-2:2002+A1:2008 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/33, IT - Security techniques.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

ISBN 978 0 580 70005 7

ICS 35.040

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2011.

Amendments issued since publication

Date	Text affected

BS ISO/IEC 9796-2:2010

INTERNATIONAL STANDARD

ISO/IEC 9796-2

Third edition
2010-12-15

Information technology — Security techniques — Digital signature schemes giving message recovery —

Part 2: Integer factorization based mechanisms

Technologies de l'information — Techniques de sécurité — Schémas de signature numérique rétablissant le message —

Partie 2: Mécanismes basés sur une factorisation entière

Reference number
ISO/IEC 9796-2:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9796-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This third edition cancels and replaces the second edition (ISO/IEC 9796-2:2002), which has been technically revised. It also incorporates the Amendment ISO/IEC 9796-2:2002/Amd.1:2008.

Implementations which comply with ISO/IEC 9796-2 (1st edition) and which use a hash-code of at least 160 bits in length will be compliant with ISO/IEC 9796-2 (3rd edition). Note, however, that implementations complying with ISO/IEC 9796-2 (1st edition) that use a hash-code of less than 160 bits in length will not be compliant with ISO/IEC 9796-2 (3rd edition). Implementations which comply with ISO/IEC 9796-2 (2nd edition) will be compliant with ISO/IEC 9796-2 (3rd edition).

ISO/IEC 9796 consists of the following parts, under the general title *Information technology — Security techniques — Digital signature schemes giving message recovery*:

— *Part 2: Integer factorization based mechanisms*

— *Part 3: Discrete logarithm based mechanisms*

Further parts may follow.

Introduction

Digital signature mechanisms can be used to provide services such as entity authentication, data origin authentication, non-repudiation, and integrity of data. A digital signature mechanism satisfies the following requirements.

— Given the verification key but not the signature key it shall be computationally infeasible to produce a valid signature for any message.

— Given the signatures produced by a signer, it shall be computationally infeasible to produce a valid signature on a new message or to recover the signature key.

— It shall be computationally infeasible, even for the signer, to find two different messages with the same signature.

NOTE 1 Computational feasibility depends on the specific security requirements and environment.

Most digital signature mechanisms are based on asymmetric cryptographic techniques and involve three basic operations:

— a process for generating pairs of keys, where each pair consists of a private signature key and the corresponding public verification key;

— a process that uses the signature key, called the signature process;

— a process that uses the verification key, called the verification process.

There are two types of digital signature mechanism.

— When, for a given signature key, two signatures produced for the same message are identical, the mechanism is said to be non-randomized (or deterministic); see ISO/IEC 14888-1.

— When, for a given message and signature key, each application of the signature process produces a different signature, the mechanism is said to be randomized.

The first and third of the three mechanisms specified in this part of ISO/IEC 9796 are deterministic (non-randomized), whereas the second of the three mechanisms specified is randomized.

Digital signature mechanisms can also be divided into the following two categories:

— When the whole message has to be stored and/or transmitted along with the signature, the mechanism is named a “signature mechanism with appendix” (see ISO/IEC 14888).

— When the whole message, or part of it, can be recovered from the signature, the mechanism is named a “signature mechanism giving message recovery” [see ISO/IEC 9796 (all parts)].

NOTE 2 Any signature mechanism giving message recovery, for example the mechanisms specified in ISO/IEC 9796 (all parts), can be converted to give a digital signature with appendix. This can be achieved by applying the signature mechanism to a hash-code derived as a function of the message. If this approach is employed, then all parties generating and verifying signatures must agree on this approach, and must also have a means of unambiguously identifying the hash-function to be used to generate the hash-code from the message.

The mechanisms specified in ISO/IEC 9796 (all parts) give either total or partial recovery, with the objective of reducing storage and transmission overhead. If the message is short enough, then the entire message can be

included in the signature, and recovered from the signature in the verification process. Otherwise, a part of the message can be included in the signature, and the remainder stored and/or transmitted along with the signature.

The mechanisms specified in this part of ISO/IEC 9796 use a hash-function for hashing the entire message (possibly in more than one part). ISO/IEC 10118 specifies hash-functions for digital signatures.

INTERNATIONAL STANDARD **ISO/IEC 9796-2:2010(E)**

Information technology — Security techniques — Digital signature schemes giving message recovery —

Part 2: Integer factorization based mechanisms

1 Scope

This part of ISO/IEC 9796 specifies three digital signature schemes giving message recovery, two of which are deterministic (non-randomized) and one of which is randomized. The security of all three schemes is based on the difficulty of factorizing large numbers. All three schemes can provide either total or partial message recovery.

This part of ISO/IEC 9796 specifies the method for key production for the three signature schemes. However, techniques for key management and for random number generation (as required for the randomized signature scheme) are outside the scope of this part of ISO/IEC 9796.

The first mechanism specified in this part of ISO/IEC 9796 is only applicable for existing implementations, and is retained for reasons of backward compatibility.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10118 (all parts), *Information technology — Security techniques — Hash-functions*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
capacity
positive integer indicating the number of bits available within the signature for the recoverable part of the message

3.2
certificate domain
collection of entities using public key certificates created by a single Certification Authority (CA) or a collection of CAs operating under a single security policy

3.3
certificate domain parameters
cryptographic parameters specific to a certificate domain and which are known and agreed by all members of the certificate domain

3.4
collision-resistant hash-function
hash-function satisfying the following property:
— it is computationally infeasible to find any two distinct inputs which map to the same output

[ISO/IEC 10118-1]

3.5
hash-code
string of bits which is the output of a hash-function

[ISO/IEC 10118-1]

3.6
hash-function
function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:
— for a given output, it is computationally infeasible to find an input which maps to this output;
— for a given input, it is computationally infeasible to find a second input which maps to the same output

[ISO/IEC 9797-2]

3.7
mask generation function
function which maps strings of bits to strings of bits of arbitrary specified length, satisfying the following property:
— it is computationally infeasible to predict, given one part of an output but not the input, another part of the output

3.8
message
string of bits of any length

[ISO/IEC 14888-1]

3.9
message representative
bit string derived as a function of the message and which is combined with the private signature key to yield the signature

3.10
nibble
block of four consecutive bits (half an octet)

3.11
non-recoverable part
part of the message stored or transmitted along with the signature; empty when message recovery is total

3.12
octet
string of eight bits

3.13
private key
key of an entity's asymmetric key pair which should only be used by that entity

[ISO/IEC 9798-1]

3.14
private signature key
private key which defines the private signature transformation

[ISO/IEC 9798-1]

3.15
public key
key of an entity's asymmetric key pair which can be made public

[ISO/IEC 9798-1]

3.16
public key system
⟨digital signature⟩ cryptographic scheme consisting of three functions:
— *key production*, a method for generating a key pair made up of a private signature key and a public verification key;
— *signature production*, a method for generating a signature Σ from a message representative F and a private signature key;
— *signature opening*, a method for obtaining the recovered message representative F^* from a signature Σ and a public verification key

NOTE The output of this function also contains an indication as to whether the signature opening procedure succeeded or failed.

3.17
public verification key
public key which defines the public verification transformation

[ISO/IEC 9798-1]

3.18
recoverable part
part of the message conveyed in the signature

3.19
salt
random data item produced by the signing entity during the generation of the message representative in Signature scheme 2

3.20
signature
string of bits resulting from the signature process

[ISO/IEC 14888-1]

3.21
trailer
string of bits of length one or two octets, concatenated to the end of the recoverable part of the message during message representative production

4 Symbols and abbreviated terms

For the purposes of this document, the following symbols and abbreviations apply.

NOTE In most cases upper case letters are used to represent bit strings and octet strings, whereas lower case letters are used to represent functions.

C	Octet string encoding the bit length of the recoverable part of the message (used in message representative production in Signature schemes 2 and 3).
c	The capacity of the signature scheme, i.e. the maximum number of bits available for the recoverable part of the message.
c^*	The recoverable message length, i.e. the length in bits of the recoverable part of the message ($c \geq c^*$).
D, D'	Bit strings constructed during message representative production in Signature schemes 2 and 3.
D^*, D'^*	Bit strings constructed during message recovery in Signature schemes 2 and 3.
F	Message representative (a bit string).
F^*	Recovered message representative (as output from the Signature opening step).
g	Mask generation function.
H	Hash-code computed as a function of the message M (a bit string).
H^*	Recovered hash-code as derived during the Message recovery step.
h	Collision-resistant hash-function.
k	The bit length of the modulus of the private signature key and public verification key (see Annex A).
L_h	The bit length of hash-codes produced by the hash-function h.
L_S	The bit length of the salt S.
M	Message to be signed (a bit string).
M^*	Message recovered from a signature as a result of the verification process.
M_1	Recoverable part of the message M, i.e. $M = M_1 \| \| M_2$.
M_1^*	Recovered recoverable part of the message (as generated during message recovery).
M_2	Non-recoverable part of the message M, i.e. $M = M_1 \| \| M_2$.
M_2^*	Non-recoverable part of the message, as input to the verification process.
N	Bit string constructed during message representative production in Signature schemes 2 and 3.
N^*	Bit string generated during message recovery in Signature schemes 2 and 3.
P	A string of zero bits constructed during message representative production in Signature schemes 2 and 3.
S	Salt (a bit string).
S^*	Recovered salt (a bit string).
t	The number of octets in the Trailer field (t = 1 or 2).
T	The Trailer field (a string of $8t$ bits used during message representative production).
Δ	Integer in the range 0 to 7 used in the specification of message allocation.

δ Integer in the range 0 to 7 used in the specification of Signature schemes 2 and 3.

Σ Signature (a bit string containing k-1 or k bits).

$|A|$ The bit length of the bit-string A, i.e. the number of bits in A.

$A \,||\, B$ Concatenation of bit strings A and B (in that order).

$\lceil a \rceil$ for a real number a, the smallest integer not less than a.

a mod n for integers a and n, (a mod n) denotes the (non-negative) remainder obtained when a is divided by n. Equivalently if $b = a$ mod n, then b is the unique integer satisfying:
(i) $0 \leq b < n$, and
(ii) (b-a) is an integer multiple of n.

$\oplus$ The bit-wise exclusive-or operator, as used to combine two binary strings of the same length.

5 Converting between bit strings and integers

To represent a non-negative integer x as a bit string of length l (l has to be such that $2^l > x$), the integer shall be written in its unique binary representation:

$$x = 2^{l-1}x_{l-1} + 2^{l-2}x_{l-2} + \ldots + 2x_1 + x_0$$

where $0 \leq x_i < 2$ (note that one or more leading digits will be zero if $x < 2^{l-1}$). The bit string shall be

$$x_{l-1}\, x_{l-2} \ldots x_0.$$

To represent a bit string $x_{l-1}\, x_{l-2} \ldots x_0$ (of length l) as an integer x, the inverse process shall be followed, i.e. x shall be the integer defined by

$$x = 2^{l-1}x_{l-1} + 2^{l-2}x_{l-2} + \ldots + 2x_1 + x_0.$$

6 Requirements

Users of this part of ISO/IEC 9796 are, wherever possible, recommended to adopt the second mechanism (Digital signature scheme 2). However, in environments where generation of random variables by the signer is deemed infeasible, then Digital signature scheme 3 is recommended.

Users who wish to employ a digital signature mechanism compliant with this part of ISO/IEC 9796 shall ensure that the following properties hold.

a) The message M to be signed shall be a binary string of any length, possibly empty.

b) The signature function uses a private signature key, while the verification function uses the corresponding public verification key.

   – Each signing entity shall use and keep secret its private signature key corresponding to its public verification key.

   – Each verifying entity should know the public verification key of the signing entity.

c) Use of the signature schemes specified in this part of ISO/IEC 9796 requires the selection of a collision-resistant hash-function h. Hash-functions are standardised in ISO/IEC 10118. There shall be a binding between the signature mechanism and the hash-function in use. Without such a binding, an adversary might claim the use of a weak hash-function (and not the actual one) and thereby forge a signature.

NOTE 1 There are various ways to accomplish this binding. The following options are listed in order of increasing risk.

1. Require a particular hash-function when using a particular signature mechanism. The verification process shall exclusively use that particular hash-function. ISO/IEC 14888-3 gives an example of this option where the DSA mechanism requires the use of Dedicated Hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

2. Allow a set of hash-functions and explicitly indicate the hash-function in use in the certificate domain parameters. Inside the certificate domain, the verification process shall exclusively use the hash-function indicated in the certificate. Outside the certificate domain, there is a risk arising from certification authorities (CAs) that may not adhere to the user's policy. If, for example, an external CA creates a certificate permitting other hash-functions, then signature forgery problems may arise. In such a case a misled verifier may be in dispute with the CA that produced the other certificate.

3. Allow a set of hash-functions and indicate the hash-function in use by some other method, e.g., an indication in the message or a bilateral agreement. The verification process shall exclusively use the hash-function indicated by the other method. However, there is a risk that an adversary may forge a signature using another hash-function.

NOTE 2 The 'other method' referred to in paragraph 3 immediately above could be in the form of a hash-function identifier included in the message representative F (see 8.2.2 and 9.2.3). If the hash-function identifier is included in F in this way then an attacker cannot fraudulently reuse an existing signature with the same M_1 and a different M_2, even when the verifier could be persuaded to accept signatures created using a hash-function sufficiently weak that pre-images can be found. However, as discussed in detail in [16] (see also Annex D), in this latter case and using the weak hash-function, an attacker can still find a new signature with a 'random' M_1.

NOTE 3 The attack mentioned in NOTE 2 that yields a new signature with a 'random' M_1 can be prevented by requiring the presence of a specific structure in M_1. For instance, one may impose a length limit on M_1 that is sufficiently less than the capacity of the signature scheme (see Annex D for further discussion). For digital signature schemes 2 and 3, a length limit on M_1 may also prevent an attacker from reusing existing signatures even if no hash-function identifier is included in the message representative, provided that the mask generation function g is based on the hash-function. This holds under the reasonable assumption that the weak hash-function involved is a 'general purpose' hash-function, not one designed solely for the purpose of forging a signature.

The user of a digital signature mechanism should conduct a risk assessment considering the costs and benefits of the various alternative means of accomplishing the required binding. This assessment should include an assessment of the cost associated with the possibility of a bogus signature being produced.

d) The verifier of a signature shall always have a secure independent means of determining which of the three signature schemes specified in this part of ISO/IEC 9796 have been employed to generate the signature. In addition, if Digital signature scheme 2 or 3 is being used, the signature verifier shall also have a means of determining which of the two signature production functions specified in Annex B have been used. This could, for example, be achieved by specifying the mechanism and signature production function in agreed 'domain parameters' or by including an unambiguous identifier for the signature scheme and signature production function in the signer's public key certificate. The signature production function may also be specified in an algorithm identifier associated with the signed data.

e) The digital signature schemes specified in this part of ISO/IEC 9796 each have particular options, the range of possible choices of which by the signer must be known to the verifier by a secure independent means. These options are as follows.

   — For all three digital signature schemes, the verifier must know whether trailer field option 1 or 2 is being employed.

   — For digital signature schemes 2 and 3, the verifier must know L_S, the length of the salt S.

   This could, for example, be achieved by specifying the option selection in the 'domain parameters' or by including option information in the signer's public key certificate.

7 Model for signature and verification processes

7.1 General

The model for a signature scheme giving message recovery presented here applies to all three of the schemes in this part of ISO/IEC 9796. When applied to a message M, a signature scheme of this type can provide either total or partial message recovery.

— If M is sufficiently short, then message recovery can be total because it is possible for M to be entirely included in the signature.

— If M is too long, then message recovery will be partial. In this case M shall be divided into the *recoverable part*, a string of bits of limited length to be included in the signature, and the *non-recoverable part*, a string of octets of any length to be stored and/or transmitted along with the signature.

The model is divided into three parts: a specification of the procedure for signing a message, a specification of the procedure for verifying a signature, and details of the additional aspects of signing and verifying that need to be defined in order to complete the specification of a signature scheme. Clauses 8, 9 and 10 specify these additional aspects for the three schemes defined in this part of ISO/IEC 9796.

7.2 Signing a message

7.2.1 Overview

Given a message M to be signed, three steps need to be performed to generate a signature on M, namely message allocation, recoverable string production, and signature production.

— *Message allocation* consists of the process whereby the message is divided into two parts: a recoverable part M_1 and a non-recoverable part M_2 (which may be empty). The length of the recoverable part is bounded above by the capacity c of the signature scheme, a value determined by the choice of the signature scheme and the key for the scheme. The recoverable part will be recovered from the signature during the verification process, whereas the non-recoverable part must be made available to the verifier by other means (e.g. it can be sent or stored with the signature). Hence, if the message is sufficiently short, the entire message can be allocated to the recoverable part, and the non-recoverable part will be empty.

— *Message representative production* takes as input the two parts of the message, and outputs a formatted string, known as the *message representative*, which is input to the signature production step.

— *Signature production* takes as input the message representative and the private signature key and outputs the *signature* Σ. This process is performed using a public key system.

7.2.2 Message allocation

The choice of signature scheme and key for the scheme determine the capacity c of the signature, where c must satisfy $c \geq 7$. The message M to be signed shall be divided into two parts, M_1 and M_2, as follows.

A recoverable message length c^* shall be chosen, where $c^* \leq c$, $c^* \leq |M|$, and $c^* \equiv |M| \pmod 8$. For Signature scheme 1, c^* shall be set equal to the minimum of $c - \Delta$ and $|M|$, where $\Delta = (c - |M|) \bmod 8$.

— If $|M| = c^*$ then the entire message shall be recoverable, i.e. $M_1 = M$ and M_2 shall be empty.

— If $|M| > c^*$ then M_1 shall be set equal to the left-most c^* bits of M, and M_2 shall be set equal to the remainder of M, i.e. M_2 contains $|M| - c^*$ bits.

In either case it follows that $M = M_1 || M_2$.

NOTE 1 For practical purposes, an application may wish to structure the message M to ensure that data it wants to be explicitly stored or transmitted (e.g., address information) is allocated to the non-recoverable message part M_2. However, the structure and interpretation of the message M are outside the scope of this part of ISO/IEC 9796.

NOTE 2 The method for message allocation ensures that M_2 is always a whole number of octets in length. Moreover, choosing c^* to be the minimum of $c–\Delta$ and $|M|$, where $\Delta = (c–|M|) \bmod 8$, ensures that M_1 is as long as possible subject to this constraint. Also, if M is a whole number of octets in length, i.e. if $|M|$ is an integer multiple of 8, then both M_1 and M_2 will consist of a whole number of octets.

7.2.3 Message representative production

This step takes as input the recoverable and non-recoverable parts of the message, M_1 and M_2, and outputs the message representative F. This shall be achieved using one of the methods specified in Clauses 8, 9 and 10 of this part of ISO/IEC 9796. These methods require use of a hash-function h and, in the cases of the second and third mechanisms, a mask generation function g that also uses h. The hash-function h to be used shall be selected from amongst those standardised in ISO/IEC 10118; the mask generation function g shall be set equal to the function specified in Annex C of this part of ISO/IEC 9796.

7.2.4 Signature production

This step takes as input the message representative F and the private signature key and outputs the *signature* Σ. This shall be achieved using the public key system specified in Annex B to this part of ISO/IEC 9796.

7.3 Verifying a signature

7.3.1 Overview

A signed message consists of either the signature Σ alone in the case of total recovery, or the non-recoverable part of the message M_2^* together with the signature Σ in the case of partial recovery. A signature shall be accepted if and only if the verification process is successful.

Given a signature Σ and non-recoverable message part M_2^*, three steps need to be performed to verify Σ and recover M^*, namely signature opening, message recovery and message assembly.

— *Signature opening* takes as input the signature Σ and the public verification key and outputs a recovered message representative F^* or returns an indication that verification has failed. This process is performed using a public key system.

— *Message recovery* takes as input the recovered message representative F^* and the non-recoverable part of the message M_2^*, and outputs the (recovered) recoverable part of the message M_1^*, or returns an indication that verification has failed.

— *Message assembly* consists of the process whereby the recovered message M^* is reconstituted from the (recovered) recoverable part M_1^* and the non-recoverable part M_2^* (which may be empty).

7.3.2 Signature opening

This step takes as input the signature Σ and the public verification key and either outputs a recovered message representative F^* or returns an indication that verification has failed. This shall be achieved using the public key system specified in Annex B to this part of ISO/IEC 9796.

7.3.3 Message recovery

This step takes as input the recovered message representative F^* and the non-recoverable part of the message M_2^*, and outputs the recoverable part of the message M_1^*, or returns an indication that verification has failed. This shall be achieved using one of the methods specified in Clauses 8, 9 and 10 of this part of ISO/IEC 9796. These methods require use of a hash-function and, in the case of the second and third

mechanisms, a mask generation function. The hash-function to be used shall be selected from amongst those standardised in ISO/IEC 10118; the mask generation function shall be set equal to the function specified in Annex C of this part of ISO/IEC 9796.

7.3.4 Message assembly

This step consists of the process whereby the message M^* is reconstituted from the recoverable part M_1^* and the non-recoverable part M_2^* (which may be empty). That is, the message M^* is assembled as $M^* = M_1^* || M_2^*$.

7.4 Specifying a signature scheme

The purpose of 7.4 is to define what choices need to be made to uniquely specify the signing and verification processes specified in this part of ISO/IEC 9796.

a) The message allocation and message assembly steps are uniquely defined within this part of ISO/IEC 9796.

b) One of the three options for the message representative production and message recovery steps, as defined in Clauses 8, 9 and 10 of this part of ISO/IEC 9796, must be chosen. Whichever of these three options is selected, a hash-function must also be chosen, which shall be selected from amongst those standardised in ISO/IEC 10118 subject to the constraint that the hash-code output shall contain at least 160 bits. In two of the three cases a mask generation function is additionally required, and the function to be employed is defined in Annex C of this part of ISO/IEC 9796.

c) The signature production and signature opening steps are uniquely defined within Annex B of this part of ISO/IEC 9796, up to the choice of the private signature key used in the signature production process and, in the case of Signature schemes 2 and 3 with an odd exponent, up to the choice between the basic and alternative signature and verification functions. The method to be used to generate pairs of private signature keys and public verification keys is defined in Annex B of this part of ISO/IEC 9796.

8 Digital signature scheme 1

8.1 General

Clause 8 defines the message representative production and message recovery processes for a deterministic digital signature scheme giving message recovery.

Because of possible attacks (see [5] and [6]), this scheme shall only be used in environments where operational constraints ensure that an attacker cannot obtain signatures on a large number of chosen messages.

NOTE Digital signature scheme 1 should only be used in environments where compatibility is required with systems implementing the first edition of this part of ISO/IEC 9796 (see [5] and [6]). However, Digital signature scheme 1 is only compatible with systems implementing the first edition of this part of ISO/IEC 9796 that use hash-codes of at least 160 bits.

8.2 Parameters

8.2.1 Modulus length

The private signature key in use is assumed to have a modulus of length k bits (see Annex B). This determines both c, the capacity of the signature, and the length of F, the message representative.

8.2.2 Trailer field options

In this scheme the trailer field (used as part of the construction of the Message representative) may be either one or two octets in length. The trailer shall consist of t octets (t = 1 or 2) where the rightmost nibble shall always be equal to hexadecimal 'C'. The following two options are permitted.

— Option 1 (t = 1): the trailer shall consist of a single octet; this octet shall be equal to hexadecimal 'BC'.

— Option 2 (t = 2): the trailer shall consist of two consecutive octets; the rightmost octet shall be equal to hexadecimal 'CC' and the leftmost octet shall be the hash-function identifier. The hash-function identifier indicates the hash-function in use.

The range '00' to '7F' is reserved for ISO/IEC JTC1; ISO/IEC 10118 specifies a unique identifier in that range for every standardized hash-function. The range '80' to 'FF' is reserved for proprietary use.

NOTE Use of the second option does not avoid the need for the verifier to have a secure independent means of knowing which hash-function to use to verify the signature. Whilst this was previously believed to be the case, this has been shown to be false, [16] (see also Annex D).

8.2.3 Capacity

The capacity c of the signature for this scheme is given by:

$$c = k - L_h - 8t - 4.$$

As defined in 7.2.2, the recoverable message length c^* shall satisfy:

a) $c^* = |M_1|$ in the case of total message recovery;

b) $c\text{-}7 \leq c^* \leq c$ in the case of partial recovery.

8.3 Message representative production

In this scheme message representative production involves two main steps:

— hashing the message;

— formatting.

8.3.1 Hashing the message

The message M (where $M = M_1||M_2$) shall be input to the hash-function h to obtain the hash-code H, i.e., $H = h(M)$. Note that H contains L_h bits.

8.3.2 Formatting

A string of k bits shall be constructed as follows (working from left to right):

— two bits set equal to '01'

— a single bit set equal to '0' in the case of total recovery (i.e. when $M = M_1$) and '1' in the case of partial recovery (i.e. when $|M_2| > 0$),

— $k - L_h - |M_1| - 8t - 4$ padding bits all set to '0',

— a single bit set equal to '1' (the final padding bit),

— the $|M_1|$ bits of M_1,

— the L_h bits of H, the hash-code,

— the $8t$ bits of the trailer field T.

NOTE 1 Where partial recovery is provided, M_2 is kept as short as possible subject to the constraint that it shall be a whole number of octets; in this case the number of padding bits equal to '0' will be less than eight.

The message representative F shall result from processing the above string from left to right in blocks of four consecutive bits, i.e., in nibbles, following the steps below.

1. The leftmost nibble shall remain unchanged.

2. If the leftmost nibble has its rightmost bit set to '0' then

   a) every subsequent nibble equal to '0000', if any, shall be replaced by a nibble equal to hexadecimal 'B'; it is part of the padding field.

   b) the first subsequent nibble not equal to '0000' shall be exclusive-ored with hexadecimal 'B' (i.e., '1011'); this is the nibble containing the final padding bit.

3. All subsequent bits shall remain unchanged.

   NOTE 2 This means that if the left-most nibble has its rightmost bit set to '1' (and hence there are no '0' padding bits), then no changes are made to the bit string.

4. The first bit of the resulting string (which will always be equal to '0') shall be deleted, resulting in F, a string of k-1 bits.

8.4 Message recovery

As specified in Clause 6, the verifier must know which hash-function h is in use prior to processing a signature. Hence the verifier will also know L_h.

If the rightmost octet of the recovered message representative F^*, a string of k-1 bits, is equal to

— hexadecimal 'BC', then the trailer consists of that single octet;

— hexadecimal 'CC', then the trailer consists of the rightmost two octets of F^*, where the leftmost octet is the identifier of the hash-function in use. This should be checked to determine whether it equals the hash-function in use by the verifier; if it disagrees then the signature verification has failed.

The signature Σ shall be rejected if either the trailer or the hash-function identifier (if present) cannot be interpreted. Otherwise, the verification process shall continue.

The signature Σ shall be rejected if the leftmost bit of the recovered message representative F^* is '0'.

A single '0' bit shall be adjoined at the left end of the string (resulting in a string of k bits). This string shall next be processed from left to right in blocks of four consecutive bits, i.e., in nibbles, following the steps below.

1. The leftmost nibble shall remain unchanged.

2. If the leftmost nibble has its rightmost bit set to '0' then

   a) every subsequent nibble equal to hexadecimal 'B', if any, is part of the padding field,

   b) the first subsequent nibble not equal to hexadecimal 'B' shall be exclusive-ored with hexadecimal 'B' to recover the initial value of this nibble.

3. All subsequent bits shall remain unchanged.

The location of the final (rightmost) padding bit can now be determined, and hence the total number of padding bits calculated. The third bit of the first nibble can also be processed to determine whether the signature provides partial or total recovery. In the case of partial recovery, the signature Σ shall be rejected if nine or more padding bits are present (i.e. eight or more zero padding bits). Otherwise, the verification process shall continue.

All bits up to the end of the padding field shall be removed from the left of the modified version of F^*, and the one or two-octet trailer shall be removed from the right. The remaining binary string shall be divided into two parts.

— The recovered hash-code H^* shall consist of the rightmost L_h bits.

— The recovered part of the message M_1^* shall consist of the remaining bits on the left.

The recovered message part M_1^* shall be concatenated with M_2^*, the non-recoverable part of the message, as submitted to the verification process, and submitted to the hash-function. If the result is the same as H^*, i.e. if $H^* = h(M_1^* || M_2^*)$, then the signature shall be accepted and M_1^* shall be returned; otherwise the signature shall be rejected.

9 Digital signature scheme 2

9.1 General

Clause 9 defines the message representative production and message recovery processes for a randomized digital signature scheme giving message recovery.

NOTE This signature scheme is compatible with the scheme known as IFSSR specified in IEEE P1363a, [10]. It is closely based on a scheme known as PSS-R, [3]. The message representative production method is similarly derived from the method known as EMSR3 in IEEE P1363a, [10].

9.2 Parameters

9.2.1 Modulus length

The private signature key in use is assumed to have a modulus of length k bits (see Annex B). This determines both c, the capacity of the signature, and the length of F, the message representative.

9.2.2 Salt length

A salt length L_S shall be selected. L_S shall be a positive integer ($L_S > 0$); a typical value is L_h.

9.2.3 Trailer field options

In this scheme the trailer field (used as part of the construction of the message representative) may be either one or two octets in length. The trailer shall consist of t octets ($t = 1$ or 2) where the rightmost nibble shall always be equal to hexadecimal ‘C’. The following two options are permitted.

— Option 1 ($t = 1$): the trailer shall consist of a single octet; this octet shall be equal to hexadecimal ‘BC’.

— Option 2 ($t = 2$): the trailer shall consist of two consecutive octets; the rightmost octet shall be equal to hexadecimal ‘CC’ and the leftmost octet shall be the hash-function identifier. The hash-function identifier indicates the hash-function in use.

The range ‘00’ to ‘7F’ is reserved for ISO/IEC JTC1; ISO/IEC 10118 specifies a unique identifier in that range for every standardized hash-function. The range ‘80’ to ‘FF’ is reserved for proprietary use.

9.2.4 Capacity

The capacity c of the signature for this scheme is given by:

$$c = k - L_h - L_S - 8t - 2.$$

9.3 Message representative production

In this scheme message representative production involves two main steps:

— hashing the message;

— formatting.

9.3.1 Hashing the message

The hash-code H shall be computed using the following or an equivalent sequence of steps.

Convert the bit length of M_1, i.e. $|M_1|$, to a bit string C of length 64 bits using the convention described in Clause 5.

Generate a fresh, random, bit string S of length L_S bits.

Compute the hash-code H as $H = h(C \,||\, M_1 \,||\, h(M_2) \,||\, S)$. Note that H contains L_h bits.

9.3.2 Formatting

The message representative F shall be computed by the following or an equivalent sequence of steps.

1. Let P be the bit string that consists of $k + \delta - L_h - L_S - |M_1| - 8t - 2$ '0' bits where $\delta = (1\text{-}k) \bmod 8$.

2. Let the bit string D be defined by $D = P \,||\, \text{'1'} \,||\, M_1 \,||\, S$, where '1' is a single bit. The length of D is $k + \delta - L_h - 8t - 1$ bits.

   NOTE If the hash-code is a whole number of octets in length, then the bit string D will also be a whole number of octets in length.

3. Apply the mask generation function g to the hash-code H to produce a bit string N of length $k + \delta - L_h - 8t - 1$ bits.

4. The length of $D \oplus N$ is $k + \delta - L_h - 8t - 1$ bits. Let D' be the string of $k - L_h - 8t - 1$ bits obtained by deleting the leftmost δ bits of $D \oplus N$.

5. Let $F = D' \,||\, H \,||\, T$, where T is the Trailer field of $8t$ bits. F is a string of k-1 bits.

9.4 Message recovery

If the rightmost octet of the recovered message representative F^*, a string of k-1 bits, is equal to

— hexadecimal 'BC', then the trailer consists of that single octet;

— hexadecimal 'CC', then the trailer consists of the rightmost two octets of F^*, where the leftmost octet is the identifier of the hash-function in use. This should be checked to determine whether it equals the hash-function in use by the verifier; if it disagrees then the signature verification has failed.

The signature Σ shall be rejected if either the trailer or the hash-function identifier (if present) cannot be interpreted. Otherwise, the verification process shall continue.

The recoverable message part M_1 shall next be recovered from the recovered message representative F^* and the non-recoverable part M_2 by the following or an equivalent sequence of steps.

1. Adjoin δ '0' bits to the leftmost end of F^*.

2. Let D'^* be the leftmost $k + \delta - L_h - 8t - 1$ bits of the resulting string, and H^* be the next L_h bits.

3. Apply the mask generation function g to the string H^* to produce a bit string N^* of length $k + \delta - L_h - 8t - 1$ bits.

4. Let $D^* = D'^* \oplus N^*$.

5. Set the leftmost δ bits of D^* to '0'.

6. Working from the leftmost end of D^*, search for the first '1' bit. Remove this bit and all zeros to the left of this bit, and then let S^* be the rightmost L_S bits of D^*, and $M_1{}^*$ be the remaining bits of D^*. If there is no first '1' bit, return an indication that verification has failed and stop.

7. Convert the bit length of $M_1{}^*$ to a bit string C of length 64 bits using the convention described in Clause 5.

8. If $H^* = h(C \,||\, M_1{}^* \,||\, h(M_2{}^*) \,||\, S^*)$ output the recovered message part $M_1{}^*$. Otherwise, return an indication that verification has failed.

10 Digital signature scheme 3

Clause 10 defines the message representative production and message recovery processes for a deterministic digital signature scheme giving message recovery.

This scheme is identical to that defined in Clause 9 with the exception that S is a fixed value which is permitted to have zero length, i.e. $L_S \geq 0$ (unlike the constraint $L_S > 0$ which applies in Clause 9). Hence this scheme is deterministic and not randomized.

The fixed salt S may be selected by the signer. Alternatively, it may be specified as part of the domain parameters.

NOTE 1 The security of this scheme is of a similar level to that obtainable from the use of 'full domain hashing', [1], [4].

NOTE 2 Digital signature scheme 3 is deemed to be preferable to Digital signature scheme 1 – see Clause 1. This is for the following reasons.

- Schemes very similar to Digital signature scheme 3 have mathematical proofs of security (see [4]). However, these proof techniques do not apply to Digital signature scheme 1.
- The two schemes have comparable efficiencies.

Annex A
(normative)

ASN.1 module

A.1 General

```
MessageRecoverySignatureMechanisms {
   iso(1) standard(0) signature-schemes(9796) part2(2) asn1-module(1)
      message-recovery-signature-mechanisms(0) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN

IMPORTS

   HashFunctions
      FROM DedicatedHashFunctions {
         iso(1) standard(0) hash-functions(10118) part(3)
            asn1-module(1) dedicated-hash-functions(0) } ;


SignatureWithMessageRecovery ::= SEQUENCE {
   algorithm  ALGORITHM.&id({MessageRecovery}),
   parameters ALGORITHM.&Type({MessageRecovery}{@algorithm}) OPTIONAL
}

MessageRecovery ALGORITHM ::= {
   dswmr-mechanism1A      |
   dswmr-mechanism2A      |
   dswmr-mechanism3A      |
   dswmr-mechanism1N      |
   dswmr-mechanism2N      |
   dswmr-mechanism3N      |
   dswmr-mechanism1A-sha1 |
   dswmr-mechanism2A-sha1 |
   dswmr-mechanism3A-sha1 |
   dswmr-mechanism1N-sha1 |
   dswmr-mechanism2N-sha1 |
   dswmr-mechanism3N-sha1,

   ...   -- Expect additional signature scheme objects --
}

dswmr-mechanism1A ALGORITHM ::= {
   OID mechanism1A PARMS HashFunctions
}

dswmr-mechanism2A ALGORITHM ::= {
   OID mechanism2A PARMS HashFunctions
}

dswmr-mechanism3A ALGORITHM ::= {
   OID mechanism3A PARMS HashFunctions
}

dswmr-mechanism1N ALGORITHM ::= {
   OID mechanism1N PARMS HashFunctions
```

```
}

dswmr-mechanism2N ALGORITHM ::= {
   OID mechanism2N PARMS HashFunctions
}

dswmr-mechanism3N ALGORITHM ::= {
   OID mechanism3N PARMS HashFunctions
}

dswmr-mechanism1A-sha1 ALGORITHM ::= { OID  mechanism1A-sha1 }

dswmr-mechanism2A-sha1 ALGORITHM ::= { OID  mechanism2A-sha1 }

dswmr-mechanism3A-sha1 ALGORITHM ::= { OID  mechanism3A-sha1 }

dswmr-mechanism1N-sha1 ALGORITHM ::= { OID  mechanism1N-sha1 }

dswmr-mechanism2N-sha1 ALGORITHM ::= { OID  mechanism2N-sha1 }

dswmr-mechanism3N-sha1 ALGORITHM ::= { OID  mechanism3N-sha1 }

-- Cryptographic algorithm identification --

ALGORITHM ::= CLASS {
   &id    OBJECT IDENTIFIER  UNIQUE,
   &Type  OPTIONAL
}
  WITH SYNTAX { OID &id [PARMS &Type] }

-- Message recovery signature mechanisms --

OID ::= OBJECT IDENTIFIER  -- Alias

signatureMechanismA OID ::= {
   iso(1)     standard(0)     signature-schemes(9796)     part2(2)     mechanism(0)
alternate(0) }

mechanism1A OID ::= { signatureMechanismA mechanism1(0) }

mechanism2A OID ::= { signatureMechanismA mechanism2(1) }

mechanism3A OID ::= { signatureMechanismA mechanism3(2) }

signatureMechanismN OID ::= {
   iso(1) standard(0) signature-schemes(9796) part2(2) mechanism(0) normal(1) }

mechanism1N OID ::= { signatureMechanismN mechanism1(0) }

mechanism2N OID ::= { signatureMechanismN mechanism2(1) }

mechanism3N OID ::= { signatureMechanismN mechanism3(2) }

-- Combined signature scheme and hash-function mechanisms --

mechanismA-Hash OID ::= {
   iso(1) standard(0) signature-schemes(9796) part2(2)
      mechanismHash(2) alternate(0) }

mechanism1A-sha1 OID ::= { mechanismA-Hash mechanism1-SHA1(0) }
```

```
mechanism2A-sha1 OID ::= { mechanismA-Hash mechanism2-SHA1(1) }

mechanism3A-sha1 OID ::= { mechanismA-Hash mechanism3-SHA1(2) }

mechanismN-Hash OID ::= {
   iso(1) standard(0) signature-schemes(9796) part2(2)
      mechanismHash(2) normal(1) }

mechanism1N-sha1 OID ::= { mechanismN-Hash mechanism1-SHA1(0) }

mechanism2N-sha1 OID ::= { mechanismN-Hash mechanism2-SHA1(1) }

mechanism3N-sha1 OID ::= { mechanismN-Hash mechanism3-SHA1(2) }

END  -- MessageRecoverySignatureMechanisms –
```

A.2 Use of subsequent object identifiers

Each of the signature schemes uses a hash-function, a sequence containing a hash algorithm identifier and any associated parameters. Therefore, the signature scheme object identifier may be followed by one of the dedicated hash-function algorithm identifiers specified in ISO/IEC 10118-3 and any associated parameters.

Using the ASN.1 XML value notation, a value of type SignatureWithMessageRecovery using normal signature processing mechanism 1 defined in this part of ISO/IEC 9796 and the SHA-1 hash-function defined in ISO/IEC 10118-3 would be represented as:

```
<SignatureWithMessageRecovery>
   <algorithm> 1.0.9796.2.0.1.0 </algorithm>
   <parameters>
      <HashFunctions>
         <algorithm> 1.3.14.3.2.26 </algorithm>
         <parameters/>
      </HashFunctions>
   </parameters>
</SignatureWithMessageRecovery>
```

A value of type SignatureWithMessageRecovery using the combined object identifier for normal signature processing mechanism 1 and the SHA-1 hash-function has the simpler form:

```
<SignatureWithMessageRecovery>
   <algorithm> 1.0.9796.2.2.1.0 </algorithm>
</SignatureWithMessageRecovery>
```

Annex B
(normative)

Public key system for digital signature

In this annex a public key system is defined. This public key system has three main parts:

— *Key production*, a method for generating a key pair made up of a private signature key and a public verification key,

— *Signature production*, a method for generating a signature Σ from a message representative *F* and a private signature key, and

— *Signature opening*, a method for obtaining the recovered message representative *F** from a signature Σ and a public verification key. The output of this function also contains an indication as to whether the signature opening procedure succeeded or failed.

B.1 Terms and definitions

For the purposes of this annex, the following terms and definitions apply.

B.1.1
modulus
integer equal to the product of two primes, and which constitutes part of the public and private keys

B.1.2
private signature key
modulus and private signature exponent

B.1.3
public verification key
modulus and public verification exponent

B.2 Symbols and abbreviations

For the purposes of this annex, and in addition to the symbols defined in Clause 4, the following symbols and abbreviations apply.

f the integer having *F* as its binary representation

*f** an integer calculated during signature opening

J an integer calculated during signature production

*J** an integer calculated during signature opening

n the modulus (part of the private signature key and the public verification key)

p, q prime factors of the modulus

s signature exponent

v verification exponent

lcm(a, b) the least common multiple of integers a and b

min{a,b} the smaller of the two values a and b.

(a | n) Jacobi symbol of a with respect to n

NOTE 1 Let p be an odd prime, and let a be a positive integer. The following formula defines the Legendre symbol of a with respect to p.

$$(a \mid p) = a^{(p-1)/2} \bmod p$$

The Legendre symbol of multiples of p with respect to p is 0. When a is not a multiple of p, the Legendre symbol of a with respect to p is either +1 or −1 depending on whether a is or is not a square modulo p.

NOTE 2 Let n be an odd positive integer, and let a be a positive integer. The Jacobi symbol of a with respect to n is the product of the Legendre symbols of a with respect to the prime factors of n (repeating the Legendre symbols for repeated prime factors). Therefore if $n = p\,q$, then $(a \mid n) = (a \mid p)\,(a \mid q)$. The Jacobi symbol of a with respect to n may be efficiently computed without knowledge of the prime factors of n.

B.3 Key production

NOTE No method is specified in this document for public-key validation, that is providing assurance to a party (that is, the party generating the key pair, the party using the public key, or a neutral third party) that a candidate public key conforms to the arithmetic definition of a public key. An invalid public key might exist because of an inadvertent key generation calculation error or the deliberate action of an adversary. Use of an invalid public key should be assumed to void all security assurances, including the inability of an adversary to forge a signature or discover the associated private key. Users that desire assurance of the arithmetic validity of a public key before using it should use other methods, such as those in ISO/IEC 9796-3. As a general principle for any cryptosystem, the use of an improperly generated but otherwise valid public key (for instance, one produced from an insufficiently random source), or an improperly protected private key, may also void all security assurances. Implementation validation can mitigate these risks as well as the possibility that invalid keys are used. However, it does not provide specific assurance that a given public key is in fact valid.

B.3.1 Public verification exponent

Each signing entity shall select a positive integer v as its public verification exponent. The public verification exponent may be standardized in specific applications.

NOTE The values 2, 3, 17 and 65537 may have some practical advantages.

B.3.2 Secret prime factors and public modulus

Each signing entity shall secretly and randomly select two distinct large primes p and q subject to the following conditions.

— If v is odd, then p–1 and q–1 shall be coprime to v.

— If v is even, then (p–1)/2 and (q–1)/2 shall be coprime to v. Moreover, p and q shall not be congruent to each other modulo 8.

The public modulus n is set equal to the product of p and q, i.e. $n = pq$. The size of modulus n in bits determines the value of k such that

$$2^{k-1} < n \leq 2^k - 1.$$

NOTE 1 Some additional conditions on the choice of primes may be taken into account in order to prevent factorization of the modulus.

NOTE 2 Some forms of the modulus simplify the modular reduction and need less table storage. Examples of such forms are

$n = 2^{64x}-r$ of length: $k = 64x$ bits,

$n = 2^{64x}+r$ of length: $k = 64x+1$ bits,

where: $1 \le y \le 2x$ and $r < 2^{64x-8y} < 2r$.

For moduli of the form $2^{64x}-r$, the most significant $8y$ bits are equal to 1, where $8y$ is at most one quarter of the modulus length. For moduli of the form $2^{64x}+r$, the most significant bit is equal to 1 and is followed by $8y$ bits equal to 0, where $8y$ is at most one quarter of the modulus length.

B.3.3 Private signature exponent

The private signature exponent shall be any positive integer s such that sv–1 is a multiple of

— lcm(p–1, q–1) if v is odd;

— lcm(p–1, q–1)/2 if v is even.

NOTE Generally, s is the least possible value.

B.4 Signature production function

The message representative F is a string of k–1 bits, where the rightmost four bits are equal to '1100' (hexadecimal 'C'). It is the binary representation of a positive integer denoted by f.

The integer J is then defined as follows:

— if v is odd, then $J = f$,

— if v is even and (f | n) = +1, then $J = f$, and

— if v is even and (f | n) = –1, then $J = f/2$.

NOTE If v is even, then the above operation ensures that the Jacobi symbol of J with respect to n is always +1.

The signature Σ is the bit string of length k-1 bits corresponding to the integer min{J^s mod n, n–(J^s mod n)} using the convention described in Clause 5.

B.5 Signature opening function

The signature Σ is a string of k-1 bits; it is the binary representation of a positive integer less than n. This integer shall be raised to the power v modulo n to get J^*, i.e.,

$$J^* = \Sigma^v \bmod n$$

The integer f^* shall then be computed as follows.

— If v is odd and

  — if $J^* \bmod 16 = 12$, then $f^* = J^*$,

  — if $J^* \bmod 16 = n - 12 \bmod 16$, then $f^* = n - J^*$.

— If v is even and

  — if $J^* \bmod 8 = 1$, then $f^* = n - J^*$.

  — if $J^* \bmod 8 = 4$, then $f^* = J^*$,

  — if $J^* \bmod 8 = 6$, then $f^* = 2J^*$,

  — if $J^* \bmod 8 = 7$, then $f^* = 2(n - J^*)$,

The signature Σ shall be rejected in all the other cases; it shall also be rejected if $f^* \bmod 16 \neq 12$, and if f^* does not satisfy $f^* \leq 2^{k-1} - 1$.

The recovered message representative F^* is the bit string of length k-1 bits corresponding to the integer f^* using the convention described in Clause 5.

B.6 Alternative signature production function

When v is odd this function may be used as an alternative to the function in Clause B.4. It shall be used together with the signature opening function in B.7.

The message representative F is a string of $k-1$ bits, where the rightmost four bits are equal to '1100' (hexadecimal 'C'). It is the binary representation of a positive integer denoted by f.

The integer J is then defined as $J = f$.

The signature Σ is the bit string of length k bits corresponding to the integer $J^s \bmod n$ using the convention described in Clause 5.

NOTE The difference between this function and the one in Clause B.4 is that the signature Σ is always $J^s \bmod n$; no "absolute value" step is performed to select the minimum of $J^s \bmod n$ and $n - (J^s \bmod n)$.

B.7 Alternative signature opening function

When v is odd this function may be used as an alternative to the function in Clause B.5. It shall be used together with the signature production function in B.6.

The signature Σ is a string of k bits; it is the binary representation of a positive integer less than n. This integer shall be raised to the power v modulo n to get J^*, i.e.,

$$J^* = \Sigma^v \bmod n.$$

The integer f^* shall then be computed as $f^* = J^*$.

The signature Σ shall be rejected if $f^* \bmod 16 \neq 12$, and if f^* does not satisfy $f^* \leq 2^{k-1} - 1$.

The recovered message representative F^* is the bit string of length $k-1$ bits corresponding to the integer f^* using the convention described in Clause 5.

NOTE The difference between this function and the one in Clause B.5 is that the integer f^* is always the same as the integer J^*; no "disambiguation" between J^* and $n - J^*$ is necessary.

Annex C
(normative)

Mask generation function

In this annex a mask generation function based on a hash-function is defined.

NOTE This function extends the one defined as MGF1 in IEEE Std 1363-2000, [9], to allow the input and output to be bit strings. It is similar to the proposals made by Bellare and Rogaway in [2] and [3].

A mask generation function takes as input a bit string Z and the desired length of the output, L_N, and outputs a bit string N of that length.

C.1 Symbols and abbreviations

For the purposes of this annex, and in addition to the symbols defined in Clause 4, the following symbols and abbreviations apply.

L_N The length (in bits) of the output from the mask generation function g.

L_Z The length (in bits) of the octet string Z.

N The output of the mask generation function g (a bit string).

Z A bit string input to the mask generating function g.

C.2 Requirements

Use of this function requires the choice of a hash-function. This hash-function, denoted here by h, shall be set equal to the hash-function h as in Clause 6 paragraph (c). We denote the output length of h in bits by L_h.

C.3 Specification

C.3.1 Parameters

One input to the function g is the desired length in bits of the output, which is a positive integer L_N.

C.3.2 Mask generation

The bit string N shall be computed by the following or an equivalent sequence of steps.

1. If L_Z exceeds the length limitation ($2^{64} - 33$ for Dedicated Hash-functions 1 and 3 from ISO/IEC 10118-3), or if $L_N > L_h \times 2^{32}$, output "error" and stop.
2. Let N be the empty string.
3. Let $i = 0$.
    - 3.1 Convert i to a bit string C of length 32 bits using the convention described in Clause 5.
    - 3.2 Let $N := N \,||\, h(\, Z \,||\, C\,)$.
    - 3.3 Let $i := i+1$.
    - 3.4 If $i < \lceil L_N/L_h \rceil$, go to Step 3.1.
4. Output the leftmost L_N bits of N.

Annex D
(informative)

On hash-function identifiers and the choice of the recoverable length of the message

As specified in Clause 6 (Requirements), users of signature schemes specified in this part of ISO/IEC 9796 must select a collision-resistant hash-function *h*. It is important that the verifier has an unambiguous way of determining which hash-function was used to generate a signature, in order that the verification process can be carried out securely. If a malicious third party could persuade a verifier that a 'weak' hash-function had been used to generate a signature (e.g. a hash-function which lacks the one-way property), then this third party could persuade the verifier that a valid signature actually applies to a 'false' message.

The three digital signature schemes specified in this part of ISO/IEC 9796 allow a hash-function identifier to be included in the message representative *F* (see 8.2.2). If the hash-function identifier is included in *F* in this way then an attacker cannot fraudulently reuse an existing signature with the same M_1 and a different M_2, even when the verifier could be persuaded to accept signatures created using a hash-function sufficiently weak that pre-images can be found. This was thought to solve the problem referred to in the previous paragraph.

However, as discussed in detail in [16], even if a hash-function identifier is included in the message representative, other attacks are possible if a verifier can be persuaded that a 'weak' hash-function has been employed. By weak here we mean a hash-function which lacks the one way property, i.e. given a hash-code it is computationally feasible to find an input string which is mapped to this hash-code by the hash-function. (Note that it is precisely this type of weakness that first motivated the inclusion of a hash-function identifier in the message representative).

The attacks described in [16] operate in the following general way. The attacker generates 'signatures' at random, and for each such 'signature' applies the public verification function of the entity whose signature he wishes to forge, and obtains the 'recovered message representative' (this is the 'signature opening' step). The next part of the attack will vary depending on the formatting of the message representative, but essentially the attacker sees if the recovered message representative is in the correct format to correspond to a genuine signature and that the hash-function identifier in this string is the one corresponding to a weak hash-function. The odds of this happening will vary, but could be as high as 2^{-16} (and hence the attacker does not need to try too many 'random signatures' before finding one with the desired properties).

Given such a 'signature', the attacker now takes the hash-code embedded in the recovered message representative and, using the fact that the hash-function is weak, discovers a non-recoverable message part, which, when combined with the recoverable part embedded in the message representative, hashes to the desired hash-code. That is, the attacker can forge a new signature with a 'random' M_1. Hence the inclusion of a hash-function identifier in the message representative does not avoid the need for the verifier to have a secure independent means of knowing which hash-function to use to verify the signature.

This discussion also relates to the choice of the recoverable length c^* for signature schemes 2 and 3. As described in 7.2.2, c^* shall be chosen so that $c^* \leq c$, the capacity of the signature scheme. It is usually desirable to choose c^* close to c so as to maximise the length of the recoverable part of the message, and hence minimise the length of the non-recoverable part of the message. It is suggested that c^* can be chosen to be somewhat less than c (e.g. c-16, c-24 or c-80, according to the desired level of difficulty), in order to make attacks of the type described above even more difficult.

Annex E
(informative)

Examples

This annex contains a total of 12 worked examples of signature production and signature verification for the three schemes defined in this part of ISO/IEC 9796, together with two examples of key production.

Clause E.1 contains examples with public exponent equal to 3.

— E.1.1 contains an example of key production.

— E.1.2 contains three examples of signature production and verification, all of which involve total message recovery. There is one example for each of the three schemes defined in this part of ISO/IEC 9796.

— E.1.3 contains three examples of signature production and verification, all of which involve partial message recovery. There is one example for each of the three schemes defined in this part of ISO/IEC 9796.

Clause E.2 contains examples with public exponent equal to 2.

— E.2.1 contains an example of key production.

— E.2.2 contains three examples of signature production and verification, all of which involve total message recovery. There is one example for each of the three schemes defined in this part of ISO/IEC 9796.

— E.2.3 contains three examples of signature production and verification, all of which involve partial message recovery. There is one example for each of the three schemes defined in this part of ISO/IEC 9796.

E.1 Examples with public exponent 3

Clause E.1 contains examples for which the public key has exponent 3.

E.1.1 Example of key production process

The example key has a modulus of k = 1024 bits with public exponent v = 3.

p = FB961451 995C82F9 527CAAAF B3FB4254 6D00A01D 8B2BDE3D 2E7B8F7D 0C9E781E
B7FABFC8 E86E9F6D ACE3435A 9D043A99 93F3E473 D93FA888 D3577906 77A94931

q = FF0EAFCA 70585166 A8CD8E90 36E75290 2F32B863 068016B6 A89F2EA3 418882EF
6F570122 F92D2E9B EFFF7329 1818F251 BF095D6E 208F93CD CEF4767A 568AB241

The public modulus n is the product of the secret prime factors p and q. Its length is 1024 bits.

n = FAA8ED34 EEF1CE38 D29814B6 EEAA154D C060BB37 EB1A51E8 AB0398DD ADDFD334
CB9BE20C 087B1DDF 1F78A397 62B5F20A 7A730086 30913CD2 EE60183D E249DD16
9CA4EB3A E0420E51 13D73050 4A73A926 BEFBFF32 C89858DE 5E5B3899 FEC52521
04933163 625F2963 5AB8FAA7 AA14C4F3 C0DD2470 DEFCEB39 2429110A 0149A771

The private signature exponent *s* is equal to the multiplicative inverse of *v* modulo lcm(*p*–1, *q*–1).

s = 0A71B48C DF4A1342 5E1BAB87 9F471638 92AEB277 A9CBC369 B1CAD109 3C93FE22
33267EC0 805A7693 F6A506D0 F9723F6B 1A6F755A ECB0B7DE 1F440102 94186936
316AAC4B F39B37BF 6105DFA0 AEA60B82 C17306F2 179F2ED4 704D5A6F BCB141C0
C9380F5A 500823CE 67E8ED81 7F8A5100 59E9541B 498C91F4 1ABE8C10 6220E72B

E.1.2 Examples with total recovery

Three examples of signature production and verification are provided, one for each of the three schemes.

E.1.2.1 Example of signature scheme 1

This example uses dedicated hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

E.1.2.1.1 The signature process

The message to be signed is the following string of 64 ASCII-coded characters.

abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqopqrpqrs

In hexadecimal, the message *M* is the following octet string of length 64, i.e. 512 bits.

M = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273

The 160 bits of the hash-code are computed by applying SHA-1 to the 512 bits of *M*.

H = 79EA0C76 F0056373 FFD6A5AA D389DD90 8B0C0E94

An identifier in the trailer field indicates the hash function in use; ISO/IEC 10118-3 sets the identifier for dedicated hash-function 3 to the value ‘33’. Therefore, the trailer field *T* consists of the following 16 bits.

T = 33CC

The message is short enough for a total recovery. The 1024 bits of the intermediate string S_i result from concatenating the two bits of the header equal to ‘01’, the more data bit equal to ‘0’, 332 (=1024–512–160–16–4) padding bits equal to ‘0’, the border bit equal to 1, the 512 bits of M_1 (=*M*), the 160 bits of *H* and the 16 bits of the trailer field *T*. The recoverable string S_r results from replacing the 82 padding nibbles equal to ‘0’ by 82 nibbles equal to ‘B’ and also the border nibble equal to ‘1’ by a nibble equal to ‘A’.

S_r = 4BBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB
BBBBBBBB
BBBBBBBB BBBBBBBB BBBA6162 63646263 64656364 65666465 66676566 67686667
68696768 696A6869 6A6B696A 6B6C6A6B 6C6D6B6C 6D6E6C6D 6E6F6D6E 6F706E6F
70716F70 71727071 727379EA 0C76F005 6373FFD6 A5AAD389 DD908B0C 0E9433CC

The recoverable integer f_r is the unsigned positive integer represented by S_r. f_r is raised to the power *s* modulo *n*. The result is represented by a temporary unsigned positive integer *t*.

t = D6369220 6E1FE0A5 7DF603C1 E5EE6025 B4EF2E69 3E8C3C9E BA00057B 40860A35
FCA66D88 33795AC1 91191515 FE852CAD C80F315C 86142051 ED322775 9F307934
421D615F 39792C40 1319F233 CFFD18D0 12D17A02 442E5BBF B17DCFC5 654BEF59
5F500A15 365CD5D0 BD27948E C938F7C3 BA775982 472E8921 7424A74B 868B63A8

Because the above result is greater than $n/2$, the signature $\Sigma = n - t$.

$\Sigma =$ 24725B14 80D1ED93 54A210F5 08BBB528 0B718CCE AC8E1549 F1039362 6D59C8FE
CEF57483 D501C31D 8E5F8E81 6430C55C B263CF29 AA7D1C81 012DF0C8 431963E2
5A8789DB A6C8E211 00BD3E1C 7A769056 AC2A8530 8469FD1E ACDD68D4 997935C7
A543274E 2C025392 9D916618 E0DBCD30 0665CAEE 97CE6217 B00469BE 7ABE43C9

The signed message consists of the 128 octets of the signature alone because M_2 is empty.

E.1.2.1.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is raised to the power 3 modulo n, thus providing the resulting integer f_s.

$f_s =$ AEED3179 3336127D 16DC58FB 32EE5992 04A4FF7C 2F5E962C EF47DD21 F2241779
0FE02650 4CBF6223 63BE4234 FF518FA7 160D9D21 CB2AD86D 87F8B2D7 7AE176AF
343B83D2 76D7A5E7 A96BC6E5 DF073EBB 528E93C6 5B29EC70 EFEBCB2B 8F54B6B1
9421C1F2 F0ECB8F1 E84580BD 9D9DD4EE 5D69249A 395217AF 469885FD F2B573A5

Since f_s is congruent to $(n - 12)$ mod 16, it is replaced by its complement to n, i.e. the recovered integer is $f_r' = n - f_s$.

$f_r' =$ 4BBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB
BBBBBBBB BBBBBBBB BBBA6162 63646263 64656364 65666465 66676566 67686667
68696768 696A6869 6A6B696A 6B6C6A6B 6C6D6B6C 6D6E6C6D 6E6F6D6E 6F706E6F
70716F70 71727071 727379EA 0C76F005 6373FFD6 A5AAD389 DD908B0C 0E9433CC

f_r' is represented as an unsigned positive integer by the recovered string S_r'.

— The leftmost octet of S_r' is equal to '4B'; it consists of the header equal to '01', the more-data bit equal to '0' (total recovery), one padding bit equal to '0' and one padding nibble equal to 'B'; it is followed by 81 nibbles equal to 'B' and the border nibble equal to 'A'; those 42 octets are removed on the right of S_r'.

— The rightmost octet of S_r' is equal to 'CC'; therefore the trailer consists from two octets, and is equal to '33CC'; those two octets are also removed on the right of S_r'.

The hash function identifier is equal to '33'; therefore the hash function in use is dedicated hash-function 3.

The remaining string of 672 bits is divided into two parts.

— M_1^* consists of the leftmost 512 bits.

— H' consists of the rightmost 160 bits.

$M_1^* =$ 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273

$H' =$ 79EA0C76 F0056373 FFD6A5AA D389DD90 8B0C0E94

The recovered message M^* consists of M_1^* alone because message recovery is total. Another hash-code H'' is computed by applying SHA-1 to M^*.

$H'' =$ 79EA0C76 F0056373 FFD6A5AA D389DD90 8B0C0E94

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.1.2.2 Example of signature scheme 2

This example uses dedicated hash-function 1 from ISO/IEC 10118-3 (otherwise known as RIPEMD-160).

E.1.2.2.1 The signature process

In hexadecimal, the message M is the following octet string of length 48, i.e., 384 bits.

M = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210

The 160 bits of salt S are generated.

S = 436BCA99 54EC376C 96B79C95 D4B82686 F3494AD3

The 160 bits of the hash-code are computed by applying dedicated hash-function 1 to the binary string of length 768 (=64+384+160+160) that results from concatenating the 64 bits of the recoverable message length C, the 384 bits of the recoverable message part M_1 (=M), the 160 bits of the hash-code of the non-recoverable message part (which is empty) $h(M_2)$ and the 160 bits of salt S. $H = h(C \,||\, M_1 \,||\, h(M_2) \,||\, S)$.

H = 50BE9461 4DA4AF5F 8E78C269 E0DFA03E 027CE74F

An identifier in the trailer field indicates the hash function in use; ISO/IEC 10118-3 sets the identifier for dedicated hash-function 1 to the value '31'. Therefore, the trailer field T consists of the following 16 bits.

T = 31CC

The message is short enough for a total recovery. The 1024 bits of the intermediate string S_i result from concatenating the 303 (=1024–384–160–160–16–1) padding bits equal to '0', the border bit equal to 1, the 384 bits of M_1 (=M), the 160 bits of L, the 160 bits of H, and the 16 bits of the trailer field T.

S_i = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 0001FEDC BA987654 3210FEDC BA987654 3210FEDC BA987654 3210FEDC
BA987654 3210FEDC BA987654 3210FEDC BA987654 3210436B CA9954EC 376C96B7
9C95D4B8 2686F349 4AD350BE 94614DA4 AF5F8E78 C269E0DF A03E027C E74F31CC

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 848 (=1024–160–16) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8).

S_r = 7BB5D930 4572EE04 BECAE622 6939DC6D A6F19867 8B339668 B09581DA 7DC69063
CFE49956 108754DD BC3AF3FF A6F562C3 6C91DAB4 BFF8CE66 29AC5B1B 6C1B524A
49B7669B 549E678C ABDAD642 A565394D 7373C4C9 4ECADF09 08A5C00D 0511B9F6
D78039FC 7F4BD793 420A50BE 94614DA4 AF5F8E78 C269E0DF A03E027C E74F31CC

The recoverable integer f_r is the unsigned positive integer represented by S_r. f_r is raised to the power s modulo n. The result is represented by the temporary unsigned positive integer t.

t = A4958BAD DA6AB0F5 E7F544BB 1313DB93 BB733605 3678459A 31386D3A 9F0A477F
37B853DF 6BBBA87B ECAC7CD2 B19FFACD 98B40E82 0B638D5F 7DDAAE56 FF198EF6
AB1002C3 76C1FFDE 03041201 FF8E6AF9 4AFDF056 06E10E32 F3F69091 34864AEB
D983AAA2 BD725FCC A288DECE 27810D34 807956DC 78F3CFC4 EA45A8DF ADA4226C

The binary string representing the integer t as an unsigned positive integer is the signature produced by the alternative signature generation function (see Clause A.6) $\Sigma' = t$.

Because the above result is greater than $n/2$, it is replaced by its complement to n. The binary string representing that integer as an unsigned positive integer is the signature $\Sigma = n - t$.

Σ = 56136187 14871D42 EAA2CFFB DB9639BA 04ED8532 B4A20C4E 79CB2BA3 0ED58BB5

```
93E38E2C 9CBF7563 32CC26C4 B115F73C E1BEF204 252DAF73 708569E6 E3304E1F
F194E877 69800E73 10D31E4E 4AE53E2D 73FE0EDC C1B74AAB 6A64A808 CA3EDA35
2B0F86C0 A4ECC996 B8301BD9 8293B7BF 4063CD94 66091B74 39E3682A 53A58505
```

The signed message consists of the 128 octets of the signature alone because M_2 is empty.

E.1.2.2.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is raised to the power 3 modulo n, thus providing the resulting integer f_s.

f_s =
```
7EF31404 A97EE034 13CD2E94 857038E0 196F22D0 5FE6BB7F FA6E1703 301942D0
FBB748B5 F7F3C901 633DAF97 BBC08F47 0DE125D1 70986E6C C4B3BD22 762E8ACC
52ED849F 8BA3A6C4 67FC5A0D A50E6FD9 4B883A69 79CD79D5 55B5788C F9B36B2A
2D12F766 E31351D0 18AEA9E9 15B3774F 117D95F8 1C930A59 83EB0E8D 19FA75A5
```

Since f_s is congruent to $(n - 12)$ mod 16, it is replaced by its complement to n, i.e. the recovered integer is $f_r' = n - f_s$.

f_r' =
```
7BB5D930 4572EE04 BECAE622 6939DC6D A6F19867 8B339668 B09581DA 7DC69063
CFE49956 108754DD BC3AF3FF A6F562C3 6C91DAB4 BFF8CE66 29AC5B1B 6C1B524A
49B7669B 549E678C ABDAD642 A565394D 7373C4C9 4ECADF09 08A5C00D 0511B9F6
D78039FC 7F4BD793 420A50BE 94614DA4 AF5F8E78 C269E0DF A03E027C E74F31CC
```

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 848 (=1024–160–16) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' =
```
80000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 0001FEDC BA987654 3210FEDC BA987654 3210FEDC BA987654 3210FEDC
BA987654 3210FEDC BA987654 3210FEDC BA987654 3210436B CA9954EC 376C96B7
9C95D4B8 2686F349 4AD350BE 94614DA4 AF5F8E78 C269E0DF A03E027C E74F31CC
```

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as $\delta = 1$ ($\delta = (1–1024) \bmod 8$). The leftmost 37 octets of the resulting binary string are equal to '0'; it is followed by the border octet '01'; those 38 octets are removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'CC'; therefore the trailer consists from two octets, and is equal to '31CC'; those two octets are also removed on the right of S_i'.

The hash function identifier is equal to '31'; therefore the hash function in use is dedicated hash-function 1.

The remaining string of 704 bits is divided into three parts.

— M_1* consists of the leftmost 384 bits.

— S* consists of the rightmost 160 bits.

— H' consists of the rightmost 160 bits.

M_1* =
```
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210
```

S* =
```
436BCA99 54EC376C 96B79C95 D4B82686 F3494AD3
```

H' =
```
50BE9461 4DA4AF5F 8E78C269 E0DFA03E 027CE74F
```

The recovered message M^* consists of M_1^* alone because message recovery is total. Another hash-code H'' is computed by applying dedicated hash-function 1 to the binary string of length 768 (=64+384+160+160), that results from concatenating the 64 bits of the recovered message length C', the 384 bits of the recovered message M^*, the 160 bits of the hash-code of the non-recoverable message part (which is empty) $h(M_2^*)$ and the 160 bits of the recovered salt S^*. $H'' = h(C' \,||\, M_1^* \,||\, h(M_2^*) \,||\, S^*)$.

H'' = `50BE9461 4DA4AF5F 8E78C269 E0DFA03E 027CE74F`

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.1.2.3 Example of signature scheme 3

This example uses dedicated hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

E.1.2.3.1 The signature process

The message to be signed is empty, i.e. a binary string of length zero.

Because this signature scheme is of deterministic type, a zero length salt value S is selected.

The 160 bits of the hash-code H are computed by applying dedicated hash-function 3 to the binary string of length 224 (=64+160), that results from concatenating the 64 bits of the recoverable message length C and the 160 bits of the hash-code of the non-recoverable message part (which is empty) $h(M_2)$. $H = h(C \,||\, h(M_2))$.

H = `A35D1688 A60AC69F D53E4442 8BFD380E 94DB9176`

The hash function in use is implicitly known. Therefore, the trailer field T consists of a single octet.

T = `BC`

The message is short enough for a total recovery. The 1024 bits of the intermediate string S_i result from concatenating the 855 (=1024–160–8–1) padding bits equal to '0', the border bit equal to 1, the 160 bits of H, and the 8 bits of the trailer field T.

S_i =
```
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 000001A3 5D1688A6
0AC69FD5 3E44428B FD380E94 DB9176BC
```

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 856 (=1024–160–8) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8).

S_r =
```
7CCB5422 2079C84C 343B0AB1 6307273B 36359229 BD3DFDEC A9FE8054 AD1EF319
44758A67 3B7C70C2 FACB6FE9 12690EE2 6DF58975 585A78C2 723F0C71 50535C80
8F0868F6 CA94F36C FB079FBB 9126286D 5EECA3CA ACA12593 033A0D64 136A7A72
D605080A 6CF68B6D DA0AE6A3 5D1688A6 0AC69FD5 3E44428B FD380E94 DB9176BC
```

The recoverable integer f_r is the unsigned positive integer represented by S_r. f_r is raised to the power s modulo n. The result is represented by the temporary unsigned positive integer t.

t =
```
F9DD9F72 FAB4AFFC ED3B0538 C5848B27 756AC50C B2890F4C BC268D96 C5E91EE8
8E3B058F 2EF6585F EF5323CA 4E2C308C C6140CF5 F5357960 5B3BF0CC 621082EB
77F4A42D 3567355E AA151FB4 652BAFFE 58A4B310 7A064669 FD4177C8 D79F5DE5
EEC562FF A2D0F5D9 C409AEA0 D5B9F8DF 493AF2F1 8F91D828 CE32C4CC 35C13113
```

The binary string representing the integer t as an unsigned positive integer is the signature produced by the alternative signature generation function (see Clause A.6) $\Sigma' = t$.

Because the above result is greater than $n/2$, the signature $\Sigma = n - t$.

Σ = 00CB4DC1 F43D1E3B E55D0F7E 29258A26 4AF5F62B 3891429B EEDD0B46 E7F6B44C
3D60DC7C D984C57F 30257FCD 1489C17D B45EF390 3B5BC372 93242771 80395A2B
24B0470D AADAD8F2 69C2109B E547F928 66574C22 4E921274 6119C0D1 2725C73B
15CDCE63 BF8E3389 96AF4C06 D45ACC14 77A2317F 4F6B1310 55F64C3D CB88765E

The signed message consists of the 128 octets of the signature alone because M_2 is empty.

E.1.2.3.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is raised to the power 3 modulo n, thus providing the resulting integer f_s.

f_s = 7DDD9912 CE7805EC 9E5D0A05 8BA2EE12 8A2B290E 2DDC53FC 01051889 00C0E01B
872657A4 CCFEAD1C 24AD33AE 504CE328 0C7D7710 D836C410 7C210BCC 91F68096
0D9C8244 15AD1AE4 18CF9094 B94D80B9 600F5B68 1BF7334B 5B212B35 EB5AAAAE
2E8E2958 F5689DF5 80AE1404 4CFE3C4D B616849B A0B8A8AD 26F10275 25B830B5

Since f_s is congruent to $(n - 12)$ mod 16, the recovered integer is $f_r' = n - f_s$.

f_r' = 7CCB5422 2079C84C 343B0AB1 6307273B 36359229 BD3DFDEC A9FE8054 AD1EF319
44758A67 3B7C70C2 FACB6FE9 12690EE2 6DF58975 585A78C2 723F0C71 50535C80
8F0868F6 CA94F36C FB079FBB 9126286D 5EECA3CA ACA12593 033A0D64 136A7A72
D605080A 6CF68B6D DA0AE6A3 5D1688A6 0AC69FD5 3E44428B FD380E94 DB9176BC

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 856 (=1024–160–8) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' = 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 000001A3 5D1688A6
0AC69FD5 3E44428B FD380E94 DB9176BC

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as $\delta = 1$ ($\delta = (1–1024)$ mod 8). The leftmost 106 octets of the resulting binary string are equal to '0'; it is followed by the border octet '01'; those 107 octets are removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'BC'; this octet is also removed on the right of S_i'.

Because the trailer is equal to 'BC', the hash function in use is implicitly known: dedicated hash-function 3 in this example.

The remaining string of 160 bits is assumed to be the hash-code H' as there is no more data left.

H' = A35D1688 A60AC69F D53E4442 8BFD380E 94DB9176

The recovered message M' is assumed to be empty and, hence, the recovery is total. Another hash-code H'' is computed by applying SHA-1 to the binary string of length 224 (=64+160), that results from concatenating the 64 bits of the recovered message length C' and the 160 bits of the hash-code of the non-recoverable message part (which is empty) $h(M_2)$. $H = h(C' \,||\, h(M_2))$.

H'' = A35D1688 A60AC69F D53E4442 8BFD380E 94DB9176

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.1.3 Examples with partial recovery

Three examples of signature production and verification are provided, one for each of the three schemes.

E.1.3.1 Example of signature scheme 1

This example uses dedicated hash-function 1 from ISO/IEC 10118-3 (otherwise known as RIPEMD-160).

E.1.3.1.1 The signature process

This example illustrates the signature of a message of 132 octets, i.e., 1056 bits.

M = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98

The 160 bits of the hash-code are computed by applying dedicated hash-function 1 to the 1056 bits of *M*.

H = F0EA911A F528FA38 777D4B9A 58B6FDA4 2D7E1999

The hash function in use is implicitly known. Therefore, the trailer field *T* consists of the following 8 bits.

T = BC

The message is too long for being entirely recoverable by the verification process. Therefore, it is divided in two parts.

- M_1 consists of the leftmost 848 bits.
- M_2 consists of the remaining 208 bits, i.e., 26 octets.

M_1 = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDC

M_2 = BA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98

The 1024 bits of the intermediate string S_i result from concatenating the two bits of the header equal to '01', the more data bit equal to '1', four (=1024–848–160–8–4) padding bits equal to '0', the border bit equal to 1, the 848 bits of M_1, the 160 bits of *H* and the 8 bits of the trailer field *T*. The recoverable string S_r results from replacing the border nibble equal to '1' by a nibble equal to 'A'.

S_r = 6AFEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCF0 EA911AF5 28FA3877 7D4B9A58 B6FDA42D 7E1999BC

The recoverable integer f_r is the unsigned positive integer represented by S_r. f_r is raised to the power *s* modulo *n*. The result is represented by the temporary unsigned positive integer *t*.

t = C9DE5B79 67CFD8BE 506749A2 F2E5035C 9C2C5E94 3DD46838 AEF7144E A01283F0
95C35FE5 53A87553 AEADBBCE 2B9876EC 14EA5C31 EA11BCC1 F33E5161 7B4B73C2
38EB6D4C AA3DF32D 1434E846 E2E74146 E24C7171 D2A0FBED 77E37371 1444360B
962A9C27 D9CC2E15 4FE30BEC A3E20B4C 0CCF472F 70E64A9C 9FFAA56A 98BC1079

Because the above result is greater than $n/2$, the signature $\Sigma = n - t$.

Σ = 30CA91BB 8721F57A 8230CB13 FBC511F1 24345CA3 AD45E9AF FC0C848F 0DCD4F44
35D88226 B4D2A88B 70CAE7C9 371D7B1E 6588A454 467F8010 FB21C6DC 66FE6954
63B97DEE 36041B23 FFA24809 678C67DF DCAF8DC0 F5F75CF0 E677C528 EA80EF15
6E68953B 8892FB4E 0AD5EEBB 0632B9A7 B40DDD41 6E16A09C 842E6B9F 688D96F8

The signed message consists of the 128 octets of the signature Σ together with the 26 octets of the non-recoverable message M_2, i.e., only 22 octets more than the message M.

E.1.3.1.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is raised to the power 3 modulo n, thus providing the resulting integer f_s.

f_s = 8FAA107A 567B7A06 C19937FC 5633C11B AF61DE7D 52A3FDB6 9A04BC23 15697F02
BA9D0551 7004C9AD 0E79C6DC CA3F9DD8 697423CB 981AE8A0 DD613B83 49D388E4
8BA60E80 47CBBA1F 02D85395 B1FD54F4 ADFD2278 302204AC 4D5C5BDF 664ED0EE
F39454A8 C9E8D531 49BA1DB6 BF83A9FE 97E2EBF9 61B150E0 6D2B6CDC 83300DB5

Since f_s is congruent to $(n - 12)$ mod 16, it is replaced by its complement to n, i.e. the recovered integer is $f_r' = n - f_s$.

f_r' = 6AFEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCF0 EA911AF5 28FA3877 7D4B9A58 B6FDA42D 7E1999BC

f_r' is represented as unsigned positive integer by the recovered string S_r'.

— The leftmost octet of S_r' is equal to '6A'; it consists of the header equal to '01', the more-data bit equal to '1' (partial recovery), one padding bit equal to '0' and the border nibble equal to 'A'; this octet is removed on the left of S_r'.

— The rightmost octet of S_r' is equal to 'BC'; this octet is also removed on the right of S_r'.

Because the trailer is equal to 'BC', the hash function in use is implicitly known: dedicated hash-function 1 in this example.

The remaining string of 1008 bits is divided into two parts.

— M_1* consists of the leftmost 848 bits.

— H' consists of the rightmost 160 bits.

M_1* = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDC

H' = F0EA911A F528FA38 777D4B9A 58B6FDA4 2D7E1999

Because the recovery is partial, the recovered message M* consists of the concatenation of M_1* and M_2*, the recovered and non-recoverable parts respectively.

M* = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98

Another hash-code H'' is computed by applying dedicated hash-function 1 to M^*.

$H'' =$ F0EA911A F528FA38 777D4B9A 58B6FDA4 2D7E1999

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.1.3.2 Example of signature scheme 2

This example uses dedicated hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

E.1.3.2.1 The signature process

The message to be signed is the following string of 112 ASCII-coded characters.

```
abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq
opqrpqrsqrstrstustuvtuvwuvwxvwxywxyzxyzayzabzabcabcdbcde
```

In hexadecimal, the message M is the following octet string of length 112 octets, i.e., 896 bits.

$M =$ 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 61626364 62636465

The 160 bits of salt S are generated.

$S =$ 4C95C1B8 7A1DE8AC C193C14C F3147FE9 C6636078

The message is too long to be entirely recoverable by the verification process. Therefore, it is divided in two parts.

- M_1 consists of the leftmost 688 bits.
- M_2 consists of the remaining 208 bits, i.e., 26 octets.

$M_1 =$ 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 7677

$M_2 =$ 7879 7778797A 78797A61 797A6162 7A616263 61626364 62636465

The 160 bits of the hash-code are computed by applying dedicated hash-function 3 to the binary string of length 1072 (=64+688+160+160), that results from concatenating the 64 bits of the recoverable message length C, the 688 bits of the recoverable message part M_1, 160 bits of the hash-code of the non-recoverable message part $h(M_2)$, and 160 bits of salt S. $H = h(C \,||\, M_1 \,||\, h(M_2) \,||\, S)$.

$H =$ 16671F61 4F2954A8 6E51CB81 102A3D47 E2C11EBD

The hash function in use is implicitly known. Therefore, the trailer field T consists of the following 8 bits.

$T =$ BC

The 1024 bits of the intermediate string S_i result from concatenating the 7 (=1024–688–160–160–8–1) padding bits equal to '0', the border bit equal to 1, the 688 bits of M_1, the 160 bits of L, the 160 bits of H, and the 8 bits of the trailer field T.

S_i = 01616263 64626364 65636465 66646566 67656667 68666768 69676869 6A68696A
6B696A6B 6C6A6B6C 6D6B6C6D 6E6C6D6E 6F6D6E6F 706E6F70 716F7071 72707172
73717273 74727374 75737475 76747576 77757677 7876774C 95C1B87A 1DE8ACC1
93C14CF3 147FE9C6 63607816 671F614F 2954A86E 51CB8110 2A3D47E2 C11EBDBC

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 856 (=1024–160–8) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ ($\delta = (1–1024) \bmod 8$).

S_r = 390871A1 2B83F417 63782F59 BB700DD4 63C071B6 98C7D152 9B8616A9 B72DF9DE
B899BFA8 C3839DD0 903CEF72 A9C18496 64129992 6FE8D3F8 FDA1D07D 2251EAAD
34017F7D C66DD60F D3F0014D 23CA2D2E 43383F30 9724B846 2529C5FE 73205FB5
D1FCC8CF D18C687F B9E35616 671F614F 2954A86E 51CB8110 2A3D47E2 C11EBDBC

The recoverable integer f_r is the unsigned positive integer represented by S_r. f_r is raised to the power s modulo n. The result is represented by the temporary unsigned positive integer t.

t = 92ACA17F 28426177 1E4A1313 C0510483 8C3CC91C 1CB6F576 CF95090A 5FDEA51E
3C189F65 E6BA3F28 4268B4FF 2363B3B9 12D023A9 1C96541A C1F9E60E 58F6B3DA
8DEB1B69 41792AA6 341DB184 88366A5E 1E18DBBA E4A2E390 77A2B4FE 1DFB34A2
CCAD1812 C4AFFAF5 5570855A AEB685DA 2E1F124F F70F529F ED02F515 BFD572AE

The binary string representing the integer t as an unsigned positive integer is the signature produced by the alternative signature generation function (see annex B.6) $\Sigma' = t$.

Because the above result is greater than $n/2$, the signature $\Sigma = n - t$.

Σ = 67FC4BB5 C6AF6CC1 B44E01A3 2E5910CA 3423F21B CE635C71 DB6E8FD3 4E012E16
8F8342A6 21C0DEB6 DD0FEE98 3F523E51 67A2DCDD 13FAE8B8 2C66322F 8953293C
0EB9CFD1 9EC8E3AA DFB97ECB C23D3EC8 A0E32377 E3F5754D E6B8839B E0C9F07E
37E61950 9DAF2E6E 0548754C FB5E3F19 92BE1220 E7ED9899 37261BF4 417434C3

The signed message consists of the 128 octets of the signature Σ together with the 26 octets of the non-recoverable message *M2*, i.e., only 42 octets more than the message M.

E.1.3.2.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is raised to the power 3 modulo n, thus providing the resulting integer f_s.

f_s = C1A07B93 C36DDA21 6F1FE55D 333A0779 5CA04981 52528096 0F7D8233 F6B1D956
13022263 44F7800E 8F3BB424 B8F46D74 166066F3 C0A868D9 F0BE47C0 BFF7F269
68A36BBD 19D43841 3FE72F03 26A97BF8 7BC3C002 3173A098 3931729B 8BA4C56B
32966893 90D2C0E3 A0D5A491 42F563A4 97887C02 8D316A28 F9EBC927 402AE9B5

Since f_s is congruent to $(n - 12) \bmod 16$, the recovered integer is $f_r' = n - f_s$.

f_r' = 390871A1 2B83F417 63782F59 BB700DD4 63C071B6 98C7D152 9B8616A9 B72DF9DE
B899BFA8 C3839DD0 903CEF72 A9C18496 64129992 6FE8D3F8 FDA1D07D 2251EAAD
34017F7D C66DD60F D3F0014D 23CA2D2E 43383F30 9724B846 2529C5FE 73205FB5
D1FCC8CF D18C687F B9E35616 671F614F 2954A86E 51CB8110 2A3D47E2 C11EBDBC

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 856 (=1024–160–8) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' = `01616263 64626364 65636465 66646566 67656667 68666768 69676869 6A68696A`
`6B696A6B 6C6A6B6C 6D6B6C6D 6E6C6D6E 6F6D6E6F 706E6F70 716F7071 72707172`
`73717273 74727374 75737475 76747576 77757677 7876774C 95C1B87A 1DE8ACC1`
`93C14CF3 147FE9C6 63607816 671F614F 2954A86E 51CB8110 2A3D47E2 C11EBDBC`

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as δ = 1 (δ = (1–1024) mod 8). The leftmost 7 bits of the resulting binary string are equal to '0'; it is followed by the border bit '1'; that 1 octet is removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'BC'; this octet is also removed on the right of S_i'.

Because the trailer field T is equal to 'BC', the hash function in use is implicitly known: dedicated hash-function 3 in this example.

The remaining string of 1008 bits is divided into three parts.

— M_1* consists of the leftmost 688 bits.

— S* consists of the rightmost 160 bits.

— H' consists of the rightmost 160 bits.

M_1* = `61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B`
`696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273`
`71727374 72737475 73747576 74757677 75767778 7677`

S* = `4C95C1B8 7A1DE8AC C193C14C F3147FE9 C6636078`

H' = `16671F61 4F2954A8 6E51CB81 102A3D47 E2C11EBD`

Because the recovery is partial, the recovered message M* consists of the concatenation of M_1* and M_2*, the recovered and non-recoverable parts respectively.

M* = `61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B`
`696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273`
`71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61`
`797A6162 7A616263 61626364 62636465`

Another hash-code H'' is computed by applying SHA-1 to the binary string of length 1072 (=64+688+160+160), that results from concatenating the 64 bits of the recovered message length C', the 688 bits of the recovered message part M_1*, the 160 bits of the hash-code of the non-recoverable message part $h(M_2$*), and the 160 bits of the recovered salt S*. $H'' = h(C' \,||\, M_1^* \,||\, h(M_2^*) \,||\, S^*)$.

H'' = `16671F61 4F2954A8 6E51CB81 102A3D47 E2C11EBD`

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.1.3.3 Example of signature scheme 3

This example uses dedicated hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

E.1.3.3.1 The signature process

This example illustrates the signature of a message of 132 octets, i.e., 1056 bits.

M = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98

Because this signature scheme is of deterministic type, a zero length salt value *S* is selected.

The message is too long for being entirely recoverable by the verification process. Therefore, it is divided in two parts.

- M_1 consists of the leftmost 840 bits.
- M_2 consists of the remaining 216 bits, i.e., 27 octets.

M_1 = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FE

M_2 = DCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98

The 160 bits of the hash-code *H* are computed by applying dedicated hash-function 3 to the binary string of length 1064 (=64+840+160), that results from concatenating the 64 bits of the recoverable message length *C*, the 840 bits of the recoverable message part M_1, and the 160 bits of the hash-code of the non-recoverable message part $h(M_2)$. $H = h(C \,||\, M_1 \,||\, h(M_2))$.

H = E30A9CB8 F10DC3C8 1897D9E8 D394555A AC6DEC79

An identifier in the trailer field *T* indicates the hash function in use, i.e. dedicated hash-function 3; ISO/IEC 10118-3 sets the identifier for this hash-function to the value '33'. Therefore, the trailer field *T* consists of the following 16 bits.

T = 33CC

The 1024 bits of the intermediate string S_i result from concatenating the 7 (=1024–840–160–16–1) padding bits equal to '0', the border bit equal to 1, 840 bits of the recoverable message part M_1, 160 bits of the hash-code of the non-recoverable message part $h(M_2)$, and the 16 bits of the trailer field *T*.

S_i = 01FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEE30A 9CB8F10D C3C81897 D9E8D394 555AAC6D EC7933CC

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 848 (=1024–160–16) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ ($\delta = (1–1024) \bmod 8$).

S_r = 1E1F9F67 4356F609 0062DEA3 FC994589 8259AE44 7F4ACAD2 0655D646 0435C851
ED9D1754 837A1269 1886C244 DED123EB 470510BC 459AD416 99310030 C824907D
0DC63B7F 422008FA 4D68E912 0DFDD534 8FAD5056 DD7A43F1 BB38E64A EEDF14CD
2549A7F6 B1869D77 C4E5E30A 9CB8F10D C3C81897 D9E8D394 555AAC6D EC7933CC

The recoverable integer f_r is the unsigned positive integer represented by S_r. f_r is raised to the power s modulo n. Being less that $n/2$ the result is kept. The binary string representing that integer as an unsigned positive integer is the signature Σ.

Σ = 30147ECB 074705DD F33EF765 D0EE1017 D5535AB3 9A7727C4 D8D4DC42 42C693BD
1FB544EC AE2323D1 185BED05 C8AA5F69 9D3AAED4 1FC3ECF9 DF297A61 56D6BC86
5196A619 806E3FDF F8A8416D 2984EF9E 33940013 4A6D1712 2FCF0946 783AEBD4
6F11397E 66863E74 28F4542D E2AE8A30 7355633F 380F937B 308C149F 14194487

In this example the signature produced by the alternative signature generation function (see annex B.6) is also the binary string Σ, i.e. $\Sigma' = \Sigma$.

The signed message consists of the 128 octets of the signature Σ together with the 27 octets of the non-recoverable message M_2, i.e., only 23 octets more than the message M.

E.1.3.3.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is raised to the power 3 modulo n, thus providing the resulting integer f_s.

f_s = 1E1F9F67 4356F609 0062DEA3 FC994589 8259AE44 7F4ACAD2 0655D646 0435C851
ED9D1754 837A1269 1886C244 DED123EB 470510BC 459AD416 99310030 C824907D
0DC63B7F 422008FA 4D68E912 0DFDD534 8FAD5056 DD7A43F1 BB38E64A EEDF14CD
2549A7F6 B1869D77 C4E5E30A 9CB8F10D C3C81897 D9E8D394 555AAC6D EC7933CC

Since f_s is here congruent to 12 mod 16, the recovered integer is $f_r' = f_s$.

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 848 (=1024–160–16) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' = 81FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEE30A 9CB8F10D C3C81897 D9E8D394 555AAC6D EC7933CC

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8). The leftmost 7 bits of the resulting binary string are equal to '0'; it is followed by the border bit '1'; that 1 octet is removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'CC'; therefore the trailer consists from two octets, and is equal to '33CC'; those two octets are also removed on the right of S_i'.

The hash function identifier is equal to '33'; therefore the hash function in use is dedicated hash-function 3.

The remaining string of 1000 bits is divided into two parts.

— M_1* consists of the leftmost 840 bits.

— H' consists of the rightmost 160 bits.

M_1* = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FE
H' = E30A9CB8 F10DC3C8 1897D9E8 D394555A AC6DEC79

Because the recovery is partial, the recovered message M^* consists of the concatenation of M_1^* and M_2^*, the recovered and non-recoverable parts respectively.

M^* = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98

Another hash-code H'' is computed by applying SHA-1 to the binary string of length 1064 (=64+840+160), that results from concatenating the 64 bits of the recovered message length C', the 840 bits of the recovered message part M_1^*, and the 160 bits of the hash-code of the non-recoverable message part $h(M_2^*)$. $H'' = h(C' \,||\, M_1^* \,||\, h(M_2^*))$.

H'' = E30A9CB8 F10DC3C8 1897D9E8 D394555A AC6DEC79

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.2 Examples with public exponent 2

Clause E.2 contains examples for which the public key has exponent 2.

E.2.1 Example of key production process

The example key has a modulus of k = 1024 bits with public exponent v = 2. Because the public verification exponent v is even, one secret prime factor is congruent to 3 modulo 8 and the other one to 7 mod 8.

p = F69AD66B F97E4CCC B4A76FD3 1F43871D C71100CA F9256C3D BE98CC23 BEC06324
A2282D3C CFCAF00B 0E7492C0 1FB19CE5 0F73EEFD 1A08B0AE 6756E7DF 5670D69B

q = C41DB9CC D8777062 2BEA8836 1E49AFA2 B5B6CBD0 28479585 472150A1 96C65E89
C2114580 FDE60F6B E12CA9DD A370A3EA 74D33B52 8EB791A9 0FD52818 3D8F612F

The public modulus n is the product of the secret prime factors p and q. Its length is 1024 bits.

n = BCEB2EB0 2E1C8E99 99BC9603 F8F91DA6 084EA6E7 C75BD18D D0CDBEDB 21DA29F1
9E731125 9DB0D190 B1920186 A8126B58 2D13ABA6 9958763A DA8F79F1 62C7379D
6109D2C9 4AA2E041 B383A74B BF17FFCC 145760AA 8B58BE3C 00C52BA3 BD05A9D0
BE5BA503 E6721FC4 066D37A8 9BF072C9 7BABB26C F6B29633 043DB474 6F9D2175

The private signature exponent s is equal to the multiplicative inverse of v mod lcm(p–1, q–1)/2.

s = 029FB5FB 55F94917 7777F3DC 7FE703F7 A3ABC251 70FDB83E 6A02DB8A 2794CECE
05C19920 85BEE677 57CCB1CC 8972089A 1D120D0C FB04C8C0 D141FE23 5A42C453
F0883D5E 73742EB5 98435B52 B393B491 F053C59C A8950D48 CA990ADF 888C6DE4
085CEB5D 6B02AEAB BCC2D543 B4C9F995 3FE16572 2F4E0846 9AD92248 D8622DEA

E.2.2 Examples with total recovery

Three examples of signature production and verification are provided, one for each of the three schemes.

E.2.2.1 Example of signature scheme 1

This example uses dedicated hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

E.2.2.1.1 The signature process

In hexadecimal, the message M is the following octet string of length 48, i.e., 384 bits.

M = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210

The 160 bits of the hash-code are computed by applying dedicated hash-function 3 to the 384 bits of M.

H = 85DCC7FC 51371637 5A059D02 5439FCD9 25C828AC

The hash function in use is implicitly known. Therefore, the trailer field T consists of the following 8 bits.

T = BC

The message is short enough for a total recovery. The 1024 bits of the intermediate string S_i result from concatenating the two bits of the header equal to '01', the more data bit equal to '0', 468 (=1024–384–160–8–4) padding bits equal to '0', the border bit equal to 1, the 384 bits of M_1 (=M), the 160 bits of H and the 8 bits of the trailer field T. The recoverable string S_r results from replacing the 116 padding nibbles equal to '0' by the 116 nibbles equal to 'B' and also the border nibble equal to '1' by a nibble equal to 'A'.

S_r = 4BBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB
BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBAFE DCBA9876
543210FE DCBA9876 543210FE DCBA9876 543210FE DCBA9876 543210FE DCBA9876
543210FE DCBA9876 54321085 DCC7FC51 3716375A 059D0254 39FCD925 C828ACBC

The recoverable integer f_r is the unsigned positive integer represented by S_r. Because the Jacobi symbol of f_r with respect to n equal to 1, the result is kept. I_r is raised to the power s modulo n. Being less that n/2 the result is kept. The binary string representing that integer as an unsigned positive integer is the signature Σ.

Σ = 0C0C62D3 523F2DA3 972679D0 348D9A50 38E93AE3 D19E97DF 875DCC04 6B2637DB
CE7D4CCC 5967529A B96D27B6 D9B41F54 56E65EEA 328FDB7D AE6F4E7D A0CFC1CF
F8AB5A80 CC7C9B9F 487EC2B5 90CBC2F3 1AFDC5CF 9C3478B9 3C46D575 A0E08D21
D965A9C4 FCAFE356 2D64B1C3 0706AF0D 43288156 DA3FF990 CB040D5C 0863F262

The signed message consists of the 128 octets of the signature alone because M_2 is empty.

E.2.2.1.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than n/2. That integer is squared modulo n, thus providing the resulting integer f_s.

f_s = 712F72F4 7260D2DD DE00DA48 3D3D61EA 4C92EB2C 0BA015D2 1512031F 661E6E35
E2B75569 E1F515D4 F5D645CA EC56AF9C 7157EFEA DD9CBA7F 1ED3BEF2 860C9F27
0CD7C1CA 6DE847CB 5F51964C E25D6755 C0254FAB AE9E25C5 AC931AA4 E04B115A
6A299405 09B7874D B23B2722 BF287678 44957B12 F11593DE CA40DB4E A77474B9

The verification process does not involve the Jacobi symbol. Because the least significant three bits of the resulting integer f_s are equal to '001', $f_r' = n - f_s$.

f_r' = 4BBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB
BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBBBB BBBBBAFE DCBA9876
543210FE DCBA9876 543210FE DCBA9876 543210FE DCBA9876 543210FE DCBA9876
543210FE DCBA9876 54321085 DCC7FC51 3716375A 059D0254 39FCD925 C828ACBC

f_r' is represented as an unsigned positive integer by the recovered string S_r'.

— The leftmost octet of S_r' is equal to '4B'; it consists of the header equal to '01', the more-data bit equal to '0' (total recovery), one padding bit equal to '0' and one padding nibble equal to 'B'; it is followed by 115 nibbles equal to 'B' and the border nibble equal to 'A'; those 59 octets are removed on the left of S_r'.

— The rightmost octet of S_r' is equal to 'BC'; this octet is also removed on the right of S_r'.

Because the trailer field T is equal to 'BC', the hash function in use is implicitly known: dedicated hash-function 3 in this example.

The remaining string of 544 bits is divided into two parts.

— M_1* consists of the leftmost 384 bits.

— H' consists of the rightmost 160 bits.

M_1* = `FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210`

H' = `85DCC7FC 51371637 5A059D02 5439FCD9 25C828AC`

The recovered message M* consists of M_1* alone because message recovery is total. Another hash-code H'' is computed by applying SHA-1 to M*.

H'' = `85DCC7FC 51371637 5A059D02 5439FCD9 25C828AC`

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.2.2.2 Example of signature scheme 2

This example uses dedicated hash-function 1 from ISO/IEC 10118-3 (otherwise known as RIPEMD-160).

E.2.2.2.1 The signature process

The message to be signed is empty, i.e. binary string of length zero.

The 160 bits of salt S are generated.

S = `61DF870C 4890FE85 D6E3DD87 C3DCE372 3F91DB49`

The 160 bits of the hash-code are computed by applying dedicated hash-function 1 to the binary string of length 384 (=64+160+160), that results from concatenating the 64 bits of the recoverable message length C, the 160 bits of the hash-code of the non-recoverable message part $h(M_2)$ and the 160 bits of salt S. $H = h(C \,||\, h(M_2) \,||\, S)$.

H = `632E21FD 52D2B95C 5F7023DA 63DE9509 C01F6C7B`

The hash function in use is implicitly known. Therefore, the trailer field T consists of the following 8 bits.

T = `BC`

The message is empty, and hence message recovery is total. The 1024 bits of the intermediate string S_i result from concatenating the 695 (=1024–160–160–8–1) padding bits equal to '0', the border bit equal to 1, the 160 bits of S, the 160 bits of H, and the 8 bits of the trailer field T.

S_i = 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000161 DF870C48 90FE85D6 E3DD87C3 DCE3723F 91DB4963 2E21FD52
D2B95C5F 7023DA63 DE9509C0 1F6C7BBC

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 856 (=1024–160–8) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8).

S_r = 73FEAF13 EB12914A 43FE6350 22BB4AB8 188A8F3A BD8D8A9E 4AD6C355 EE920359
C7F237AE 36B1212F E947F676 C68FE362 247D27D1 F298CA93 02EB21F4 A64C26CE
44471EF8 C0DFE1A5 4606F0BA 8E63E87C DACA993B FA62973B 567473B4 D38FAE73
AB228600 934A9CC1 D3263E63 2E21FD52 D2B95C5F 7023DA63 DE9509C0 1F6C7BBC

The recoverable integer f_r is the unsigned positive integer represented by S_r. Because the Jacobi symbol of f_r with respect to n equal to –1, the representative integer is $J = f_r/2$.

J = 39FF5789 F58948A5 21FF31A8 115DA55C 0C45479D 5EC6C54F 256B61AA F74901AC
E3F91BD7 1B589097 F4A3FB3B 6347F1B1 123E93E8 F94C6549 817590FA 53261367
22238F7C 606FF0D2 A303785D 4731F43E 6D654C9D FD314B9D AB3A39DA 69C7D739
D5914300 49A54E60 E9931F31 9710FEA9 695CAE2F B811ED31 EF4A84E0 0FB63DDE

J is raised to the power s modulo n. The result is as follows.

J^s = B6935ACC DCABB323 D7A7125A CA86B2E6 AF7937DE 4F523629 93B07BF2 895A4677
50553ECE 92570E7F 975CDB89 D3EC9487 CA626E9B 4E7FD5A4 16ED9C7A 9E619DCF
DC05A5A9 4089E593 50C9E86B 4DD10E5B DD709150 843D755B 057C99F6 71330258
E56474B9 6A7A4848 DC1F4100 1603BBAB DBA44AE7 1A6F8211 40137572 67C97D0C

Because the above result is greater than $n/2$, the signature $\Sigma = n - t$.

Σ = 0657D3E3 5170DB75 C21583A9 2E726ABF 58D56F09 78099B64 3D1D42E8 987FE37A
4E1DD257 0B59C311 1A3525FC D425D6D0 62B13D0B 4AD8A096 C3A1DD76 C46599CD
85042D20 0A18FAAE 62B9BEE0 7146F170 36E6CF5A 071B48E0 FB4891AD 4BD2A777
D8F7304A 7BF7D77B 2A4DF6A8 85ECB71D A0076785 DC431421 C42A3F02 07D3A469

The signed message consists of the 128 octets of the signature alone because M_2 is empty.

E.2.2.2.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is squared modulo n, thus providing the resulting integer f_s.

f_s = 39FF5789 F58948A5 21FF31A8 115DA55C 0C45479D 5EC6C54F 256B61AA F74901AC
E3F91BD7 1B589097 F4A3FB3B 6347F1B1 123E93E8 F94C6549 817590FA 53261367
22238F7C 606FF0D2 A303785D 4731F43E 6D654C9D FD314B9D AB3A39DA 69C7D739
D5914300 49A54E60 E9931F31 9710FEA9 695CAE2F B811ED31 EF4A84E0 0FB63DDE

The verification process does not involve the Jacobi symbol. Because the least significant three bits of the resulting integer f_s are equal to '110', $f_r' = 2f_s$.

f_r' = 73FEAF13 EB12914A 43FE6350 22BB4AB8 188A8F3A BD8D8A9E 4AD6C355 EE920359
C7F237AE 36B1212F E947F676 C68FE362 247D27D1 F298CA93 02EB21F4 A64C26CE
44471EF8 C0DFE1A5 4606F0BA 8E63E87C DACA993B FA62973B 567473B4 D38FAE73
AB228600 934A9CC1 D3263E63 2E21FD52 D2B95C5F 7023DA63 DE9509C0 1F6C7BBC

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 856 (=1024–160–8) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' = `00000000 00000000 00000000 00000000 00000000 00000000 00000000`
`00000000 00000000 00000000 00000000 00000000 00000000 00000000`
`00000000 00000000 00000000 00000000 00000000 00000000 00000000`
`00000161 DF870C48 90FE85D6 E3DD87C3 DCE3723F 91DB4963 2E21FD52`
`D2B95C5F 7023DA63 DE9509C0 1F6C7BBC`

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8). The leftmost 695 bits of the resulting binary string are equal to '0'; it is followed by the border bit '1'; those 87octets are removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'BC'; this octet is also removed on the right of S_i'.

Because the trailer is equal to 'BC', the hash function in use is implicitly known: dedicated hash-function 1 in this example.

The remaining string of 320 bits is divided into two parts.

— S^* consists of the rightmost 160 bits.

— H' consists of the rightmost 160 bits.

S^* = `61DF870C 4890FE85 D6E3DD87 C3DCE372 3F91DB49`

H' = `632E21FD 52D2B95C 5F7023DA 63DE9509 C01F6C7B`

The recovered message M^* is assumed to be empty and, hence message recovery is total. Another hash-code H'' is computed by applying dedicated hash-function 1 to the binary string of length 384 (=64+160+160), that results from concatenating the 64 bits of the recoverable message length C', the 160 bits of the hash-code of the non-recoverable message part $h(M_2)$ and the 160 bits of salt S^*. $H = h(C' \,||\, h(M_2) \,||\, S^*)$.

H'' = `632E21FD 52D2B95C 5F7023DA 63DE9509 C01F6C7B`

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.2.2.3 Example of signature scheme 3

This example uses dedicated hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

E.2.2.3.1 The signature process

The message to be signed is the following string of 64 ASCII-coded characters.

`abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqopqrpqrs`

In hexadecimal, the message M is the following octet string of length 64, i.e., 512 bits.

M = `61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B`
`696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273`

Because this signature scheme is of deterministic type, a zero length salt value is selected.

The 160 bits of the hash-code H are computed by applying dedicated hash-function 3 to the binary string of length 736 (= 64+512+160), that results from concatenating the 64 bits of the recoverable message length C, the 512 bits of the recoverable message part M_1, the 160 bits of the hash-code of the non-recoverable message part $h(M_2)$, which is empty. $H = h(C \,||\, M_1 \,||\, h(M_2))$.

$H =$ `D74009C4 638462E6 9D5923E7 433AEC02 8B9A90E6`

An identifier in the trailer field indicates the hash function in use; ISO/IEC 10118-3 sets the identifier for dedicated hash-function 3 to the value '33'. Therefore, the trailer field T consists of the following 16 bits.

$T =$ `33CC`

The message is short enough for a total recovery. The 1024 bits of the intermediate string S_i result from concatenating the 335 (=1024–512–160–16–1) padding bits equal to '0', the border bit equal to 1, the 512 bits of M_1 (=M), the 160 bits of S, the 160 bits of H, and the 16 bits of the trailer field T.

$S_i =$
```
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00016162 63646263 64656364 65666465
66676566 67686667 68696768 696A6869 6A6B696A 6B6C6A6B 6C6D6B6C
6D6E6C6D 6E6F6D6E 6F706E6F 70716F70 71727071 7273D740 09C46384
62E69D59 23E7433A EC028B9A 90E633CC
```

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 848 (=1024–160–16) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ ($\delta = (1–1024) \bmod 8$).

$S_r =$
```
296B0622 4010E1EC 230D4560 A5F88F03 550AAFCE 31C805CE 81E811E5 E53E5F71
AE64FC2A 2A486B19 3E87972D 90C54B80 7A862F21 A21919A4 3ECF0672 40A8C8C6
41DE8DCD F1942CF7 90D13672 8FFC0D98 FB906E79 39C1EC0E 64C0E067 F0A7443D
6170E411 DF91F797 D1FFD740 09C46384 62E69D59 23E7433A EC028B9A 90E633CC
```

The recoverable integer f_r is the unsigned positive integer represented by S_r. Because the Jacobi symbol of f_r with respect to n equal to –1, the representative integer is $J = f_r/2$.

$J =$
```
14B58311 200870F6 1186A2B0 52FC4781 AA8557E7 18E402E7 40F408F2 F29F2FB8
D7327E15 1524358C 9F43CB96 C862A5C0 3D431790 D10C8CD2 1F678339 20546463
20EF46E6 F8CA167B C8689B39 47FE06CC 7DC8373C 9CE0F607 32607033 F853A21E
B0B87208 EFC8FBCB E8FFEBA0 04E231C2 31734EAC 91F3A19D 760145CD 487319E6
```

J is raised to the power s modulo n. Because the result is less that $n/2$, the signature $\Sigma = J^s$.

$\Sigma =$
```
4F9FE3FA 21E8EAE7 786363CC D14D0AE6 401174BC B94AFBE8 3E24D014 4CB8CDF1
075E4D92 F4E08091 7DFC3C66 3A65457A 3178F280 DFF7E16C A9D29BCD B18AE2AE
C483A97F 2EF1FB4C 7BBFA1D1 269BFAF5 245C27DA E6DF3531 CADEE605 74A97378
21454089 91530D1B F8AED104 CB95149B 28E552DB 1A611286 2C099D7A 442A462B
```

The signed message consists of the 128 octets of the signature alone because M_2 is empty.

E.2.2.3.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is raised squared modulo n, thus providing the resulting integer f_s.

$f_s =$
```
14B58311 200870F6 1186A2B0 52FC4781 AA8557E7 18E402E7 40F408F2 F29F2FB8
D7327E15 1524358C 9F43CB96 C862A5C0 3D431790 D10C8CD2 1F678339 20546463
20EF46E6 F8CA167B C8689B39 47FE06CC 7DC8373C 9CE0F607 32607033 F853A21E
B0B87208 EFC8FBCB E8FFEBA0 04E231C2 31734EAC 91F3A19D 760145CD 487319E6
```

The verification process does not involve the Jacobi symbol. Because the least significant three bits of the resulting integer f_s are equal to '110', $f_r' = 2f_s$.

f_r' = 296B0622 4010E1EC 230D4560 A5F88F03 550AAFCE 31C805CE 81E811E5 E53E5F71
AE64FC2A 2A486B19 3E87972D 90C54B80 7A862F21 A21919A4 3ECF0672 40A8C8C6
41DE8DCD F1942CF7 90D13672 8FFC0D98 FB906E79 39C1EC0E 64C0E067 F0A7443D
6170E411 DF91F797 D1FFD740 09C46384 62E69D59 23E7433A EC028B9A 90E633CC

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 848 (=1024–160–16) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' = 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00016162 63646263 64656364 65666465
66676566 67686667 68696768 696A6869 6A6B696A 6B6C6A6B 6C6D6B6C
6D6E6C6D 6E6F6D6E 6F706E6F 70716F70 71727071 7273D740 09C46384
62E69D59 23E7433A EC028B9A 90E633CC

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8). The leftmost 335 bits of the resulting binary string are equal to '0'; it is followed by the border bit '1'; those 42octets are removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'CC'; therefore the trailer consists from two octets, and is equal to '33CC'; those two octets are also removed on the right of S_i'.

The hash function identifier is equal to '33'; therefore the hash function in use is dedicated hash-function 3.

The remaining string of 672 bits is divided into two parts.

— M_1* consists of the leftmost 512 bits.

— H' consists of the rightmost 160 bits.

M_1* = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273

H' = D74009C4 638462E6 9D5923E7 433AEC02 8B9A90E6

The recovered message M* consists of M_1* alone because message recovery is total. Another hash-code H'' is computed by applying SHA-1 to the binary string of length 736 (=64+512+160), that results from concatenating the 64 bits of the recovered message length C', the 512 bits of the recovered message part M_1*, the 160 bits of the hash-code of the (empty) non-recoverable message part $h(M_2^*)$. $H'' = h(C' \,||\, M_1^* \,||\, h(M_2^*))$.

H'' = D74009C4 638462E6 9D5923E7 433AEC02 8B9A90E6

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.2.3 Examples with partial recovery

Three examples of signature production and verification are provided, one for each of the three schemes.

E.2.3.1 Example of signature scheme 1

This example uses dedicated hash-function 3 from ISO/IEC 10118-3 (otherwise known as SHA-1).

E.2.3.1.1 The signature process

The message to be signed is the following string of 112 ASCII-coded characters.

```
abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq
opqrpqrsqrstrstustuvtuvwuvwxvwxywxyzxyzayzabzabcabcdbcde
```

In hexadecimal, the message *M* is the following octet string of length 112 octets, i.e., 896 bits.

M = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 61626364 62636465

The 160 bits of the hash-code are computed by applying SHA-1 to the 896 bits of *M*.

H = 1CF7A997 4518E555 C1802CB8 10A23C27 4FCFAA73

An identifier in the trailer field indicates the hash function in use; ISO/IEC 10118-3 sets the identifier for dedicated hash-function 3 to the value '33'. Therefore, the trailer field *T* consists of the following 16 bits.

T = 33CC

The message is too long to be entirely recoverable by the verification process. Therefore, it is divided into two parts.

- M_1 consists of the leftmost 840 bits.
- M_2 consists of the remaining 56 bits, i.e. 7 octets.

M_1 = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 61
M_2 = 626364 62636465

The 1024 bits of the intermediate string S_i result from concatenating the two bits of the header equal to '01', the more data bit equal to '1', four (=1024–840–160–16–4) padding bits equal to '0', the border bit equal to 1, the 840 bits of M_1, the 160 bits of *H* and the 16 bits of the trailer field *T*. The recoverable string S_r results from replacing the border nibble equal to '1' by a nibble equal to 'A'.

S_r = 6A616263 64626364 65636465 66646566 67656667 68666768 69676869 6A68696A
6B696A6B 6C6A6B6C 6D6B6C6D 6E6C6D6E 6F6D6E6F 706E6F70 716F7071 72707172
73717273 74727374 75737475 76747576 77757677 78767778 79777879 7A78797A
61797A61 627A6162 63611CF7 A9974518 E555C180 2CB810A2 3C274FCF AA7333CC

The recoverable integer f_r is the unsigned positive integer represented by S_r. Because the Jacobi symbol of f_r with respect to *n* equal to –1, the representative integer is $J = f_r/2$.

J = 3530B131 B23131B2 32B1B232 B33232B3 33B2B333 B43333B4 34B3B434 B53434B5
35B4B535 B63535B6 36B5B636 B73636B7 37B6B737 B83737B8 38B7B838 B93838B9
39B8B939 BA3939BA 3AB9BA3A BB3A3ABB 3BBABB3B BC3B3BBC 3CBBBC3C BD3C3CBD
30BCBD30 B13D30B1 31B08E7B D4CBA28C 72AAE0C0 165C0851 1E13A7E7 D53999E6

J is raised to the power *s* modulo *n*. The result is as follows.

J^s = AD83029D FB27EC14 E5F8FDC0 8E10481F 35CB879F 62BC2180 A17D84DE 9C65FF91
728DEAC0 6848885A AC99863F 0B937046 2FF8C5ED 33DA98CB E75D54BA 59CC12BF
E9AF94B7 80E31542 A96FD25A 4B1AD996 0032373A 208D4965 0870EEB0 A771F302
74B08389 95F0B131 F0CE526F 679B1618 A1BCAAAB 45AE4669 421339D3 6398C111

Because the above result is greater than $n/2$, the signature $\Sigma = n - J^s$.

Σ = 0F682C12 32F4A284 B3C39843 6AE8D586 D2831F48 649FB00D 2F5039FC 85742A60
2BE52665 35684936 04F87B47 9C7EFB11 FD1AE5B9 657DDD6E F3322537 08FB24DD
775A3E11 C9BFCAFF 0A13D4F1 73FD2636 14252970 6ACB74D6 F8543CF3 1593B6CE
49AB217A 50816E92 159EE539 34555CB0 D9EF07C1 B1044FC9 C22A7AA1 0C046064

The signed message consists of the 128 octets of the signature Σ together with the 7 octets of the non-recoverable part M_2, i.e. only 23 octets more than the message M.

E.2.3.1.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is squared modulo n, thus providing the resulting integer f_s.

f_s = 3530B131 B23131B2 32B1B232 B33232B3 33B2B333 B43333B4 34B3B434 B53434B5
35B4B535 B63535B6 36B5B636 B73636B7 37B6B737 B83737B8 38B7B838 B93838B9
39B8B939 BA3939BA 3AB9BA3A BB3A3ABB 3BBABB3B BC3B3BBC 3CBBBC3C BD3C3CBD
30BCBD30 B13D30B1 31B08E7B D4CBA28C 72AAE0C0 165C0851 1E13A7E7 D53999E6

The verification process does not involve the Jacobi symbol. Because the least significant three bits of the resulting integer f_s are equal to '110', $f_r' = 2f_s$.

f_r' = 6A616263 64626364 65636465 66646566 67656667 68666768 69676869 6A68696A
6B696A6B 6C6A6B6C 6D6B6C6D 6E6C6D6E 6F6D6E6F 706E6F70 716F7071 72707172
73717273 74727374 75737475 76747576 77757677 78767778 79777879 7A78797A
61797A61 627A6162 63611CF7 A9974518 E555C180 2CB810A2 3C274FCF AA7333CC

f_r' is represented as an unsigned positive integer by the recovered string S_r'.

— The leftmost octet of S_r' is equal to '6A'; it consists of the header equal to '01', the more-data bit equal to '1' (partial recovery), one padding bit equal to '0' and the border nibble equal to 'A'; this octet is removed on the left of S_r'.

— The rightmost octet of S_r' is equal to 'CC'; therefore the trailer consists from two octets, and is equal to '33CC'; those two octets are also removed on the right of S_r'.

The hash function identifier is equal to '33'; therefore the hash-function in use is dedicated hash-function 3.

The remaining string of 1000 bits is divided into two parts.

— M_1* consists of the leftmost 840 bits.

— H' consists of the rightmost 160 bits.

M_1* = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 61

H' = 1CF7A997 4518E555 C1802CB8 10A23C27 4FCFAA73

Because the recovery is partial, the recovered message M^* consists of the concatenation of M_1^* and M_2^*, the recovered and non-recoverable parts respectively.

M^* = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 61626364 62636465

Another hash-code H'' is computed by applying dedicated hash-function 3 to the recovered message M^*.

H'' = 1CF7A997 4518E555 C1802CB8 10A23C27 4FCFAA73

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.2.3.2 Example of signature scheme 2

This example uses dedicated hash-function 1 from ISO/IEC 10118-3 (otherwise known as RIPEMD-160).

E.2.3.2.1 The signature process

This example illustrates the signature of a message of length 132 octets, i.e., 1056 bits.

M = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98

The 160 bits of salt S are generated.

S = 78E29320 3CBA1B7F 92F05F4D 171FF8CA 3E738FF8

The message is too long for being entirely recoverable by the verification process. Therefore, it is divided in two parts.

— M_1 consists of the leftmost 680 bits.

— M_2 consists of the remaining 376 bits, i.e., 47 octets.

M_1 = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76

M_2 = 543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98

The 160 bits of the hash-code are computed by applying dedicated hash-function 1 to the binary string of length 1064 (=64+680+160+160), that results from concatenating the 64 bits of the recoverable message length C, the 680 bits of the recoverable message part M_1, the 160 bits of the hash-code of the non-recoverable message part $h(M_2)$, and the 160 bits of salt S. $H = h(C \,||\, M_1 \,||\, h(M_2) \,||\, S)$.

H = A4B517F2 E820B81F 26BCE7C6 6F48A2DB 12A8F3D7

An identifier in the trailer indicates the hash function in use; ISO/IEC 10118-3 sets the identifier for dedicated hash-function 1 to the value ‘31’. Therefore, the trailer field T consists of the following 16 bits.

T = 31CC

The 1024 bits of the intermediate string S_i result from concatenating the 7 (=1024–680–160–160–16–1) padding bits equal to '0', the border bit equal to 1, the 680 bits of M_1, the 160 bits of L, the 160 bits of H, and the 16 bits of the trailer field T.

S_i = 01FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 987678E2 93203CBA 1B7F92F0
5F4D171F F8CA3E73 8FF8A4B5 17F2E820 B81F26BC E7C66F48 A2DB12A8 F3D731CC

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 848 (=1024–160–16) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8).

S_r = 01402B29 ABA10407 9677CE7F C3D5A84D B24494D6 F9508B45 96484F5B 3CC7E8AF
CC4DDE70 81F21CAE 9D4F94D6 D2CCCB43 FCEDA098 8FFD4EF2 EAE72CFD EB4A2638
F0A34A0C 49664CD9 DB723315 759D7588 36C8BA26 AC4348B6 6958AC94 AE0B5A75
195B57AB FB9971E2 1337A4B5 17F2E820 B81F26BC E7C66F48 A2DB12A8 F3D731CC

The recoverable integer f_r is the unsigned positive integer represented by S_r. Because the Jacobi symbol of f_r with respect to n equal to –1, the representative integer is $J = f_r/2$.

J = 00A01594 D5D08203 CB3BE73F E1EAD426 D9224A6B 7CA845A2 CB2427AD 9E63F457
E626EF38 40F90E57 4EA7CA6B 696665A1 FE76D04C 47FEA779 7573967E F5A5131C
7851A506 24B3266C EDB9198A BACEBAC4 1B645D13 5621A45B 34AC564A 5705AD3A
8CADABD5 FDCCB8F1 099BD25A 8BF97410 5C0F935E 73E337A4 516D8954 79EB98E6

J is raised to the power s modulo n. The result is as follows.

J^s = 66313F1F BCE72AD5 7D32353B DAF0D50C 3915C837 D12F5C46 DFC76A59 557D27F9
B41CF256 94EB6105 6E029BD5 C4F91C20 D687FAC0 26BA47EF 05AD83DC FE5CF985
EF6B1B91 282460B8 77A8C111 2628F25A 519EF21E E2C1EB0F 019CE73C 747F435C
6DB21E45 28A96AB9 76289AB7 99D32256 4167D935 5C3F3D7F 84AE87C2 2AA23A5A

Because the above result is greater than $n/2$, the signature $\Sigma = n - J^s$.

Σ = 56B9EF90 713563C4 1C8A60C8 1E084899 CF38DEAF F62C7546 F1065481 CC5D01F7
EA561ECF 08C5708B 438F65B0 E3194F37 568BB0E6 729E2E4B D4E1F614 646A3E17
719EB738 227E7F89 3BDAE63A 98EF0D71 C2B86E8B A896D32C FF284467 48866674
50A986BE BDC8B50A 90449CF1 021D5073 3A43D937 9A7358B3 7F8F2CB2 44FAE71B

The signed message consists of the 128 octets of the signature Σ together with the 47 octets of the non-recoverable message M_n, i.e., only 43 octets more than the message M.

E.2.3.2.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is squared modulo n, thus providing the resulting integer f_s.

f_s = BC4B191B 584C0C95 CE80AEC4 170E497F 2F2C5C7C 4AB38BEB 05A9972D 83763599
B84C21ED 5CB7C339 62EA371B 3EAC05B6 2E9CDB5A 5159CEC1 651BE372 6D222480
E8B82DC3 25EFB9D4 C5CA8DC1 04494507 F8F30397 353719E0 CC18D559 65FFFC96
31ADF92D E8A566D2 FCD1654E 0FF6FEB9 1F9C1F0E 82CF5E8E B2D02B1F F5B1888F

The verification process does not involve the Jacobi symbol. Because the least significant three bits of the resulting integer f_s are equal to '111', $f_r' = 2(n - f_s)$.

f_r' = 01402B29 ABA10407 9677CE7F C3D5A84D B24494D6 F9508B45 96484F5B 3CC7E8AF
CC4DDE70 81F21CAE 9D4F94D6 D2CCCB43 FCEDA098 8FFD4EF2 EAE72CFD EB4A2638
F0A34A0C 49664CD9 DB723315 759D7588 36C8BA26 AC4348B6 6958AC94 AE0B5A75
195B57AB FB9971E2 1337A4B5 17F2E820 B81F26BC E7C66F48 A2DB12A8 F3D731CC

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 848 (=1024–160–16) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' = 01FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 98765432
10FEDCBA 98765432 10FEDCBA 98765432 10FEDCBA 987678E2 93203CBA 1B7F92F0
5F4D171F F8CA3E73 8FF8A4B5 17F2E820 B81F26BC E7C66F48 A2DB12A8 F3D731CC

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8). The leftmost 7 bits of the resulting binary string are equal to '0'; it is followed by the border bit '1'; that 1 octet is removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'CC'; therefore the trailer consists from two octets, and is equal to '31CC'; those two octets are also removed on the right of S_i'.

The hash function identifier is equal to '31'; therefore the hash function in use is dedicated hash-function 1.

The remaining string of 1000 bits is divided into three parts.

— M_1* consists of the leftmost 680 bits.

— S* consists of the rightmost 160 bits.

— H' consists of the rightmost 160 bits.

M_1* = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76

S* = 78E29320 3CBA1B7F 92F05F4D 171FF8CA 3E738FF8

H' = A4B517F2 E820B81F 26BCE7C6 6F48A2DB 12A8F3D7

Because the recovery is partial, the recovered message M* consists of the concatenation of M_1* and M_2*, the recovered and non-recoverable parts respectively.

M* = FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210 FEDCBA98 76543210
FEDCBA98

Another hash-code H'' is computed by applying dedicated hash-function 1 to the binary string of length 1064 (=64+680+160+160), that results from concatenating the 64 bits of the recovered message length C, the 680 bits of the recovered message part M_1*, the 160 bits of the hash-code of the non-recoverable message part $h(M_2^*)$, and the 160 bits of the recovered salt S*. $H'' = h(C \,||\, M_1^* \,||\, h(M_2^*) \,||\, S^*)$.

H''= A4B517F2 E820B81F 26BCE7C6 6F48A2DB 12A8F3D7

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

E.2.3.3 Example of signature scheme 3

This example uses dedicated hash-function 1 from ISO/IEC 10118-3 (otherwise known as RIPEMD-160).

E.2.3.3.1 The signature process

The message to be signed is the following string of 112 ASCII-coded characters.

```
abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq
opqrpqrsqrstrstustuvtuvwuvwxvwxywxyzxyzayzabzabcabcdbcde
```

In hexadecimal, the message M is the following octet string of length 112 octets, i.e., 896 bits.

M = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 61626364 62636465

Because this signature scheme is of deterministic type, a zero length salt value is selected.

The message is too long to be entirely recoverable by the verification process. Therefore, it is divided in two parts.

— M_1 consists of the leftmost 848 bits.

— M_2 consists of the remaining 48 bits, i.e., 6 octets.

M_1 = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 6162
M_2 = 6364 62636465

The 160 bits of the hash-code H are computed by applying dedicated hash-function 1 to the binary string of length 1072 (=64+848+160), that results from concatenating the 64 bits of the recoverable message length C, the 848 bits of the recoverable message part M_1 and the 160 bits of the hash-code of the non-recoverable message part $h(M_2)$. $H = h(C \,||\, M_1 \,||\, h(M_2))$.

H = 15F000AC 58EE3FFF 144845E7 71907C0C 83324A6F

The hash function in use is implicitly known. Therefore, the trailer field T consists of the following 8 bits.

T = BC

The 1024 bits of the intermediate string S_i result from concatenating the 7 (=1024–848–160–8–1) padding bits equal to '0', the border bit equal to 1, the 848 bits of M_r, the 160 bits of H and the 8 bits of the trailer field T.

S_i = 01616263 64626364 65636465 66646566 67656667 68666768 69676869 6A68696A
6B696A6B 6C6A6B6C 6D6B6C6D 6E6C6D6E 6F6D6E6F 706E6F70 716F7071 72707172
73717273 74727374 75737475 76747576 77757677 78767778 79777879 7A78797A
61797A61 627A6162 63616215 F000AC58 EE3FFF14 4845E771 907C0C83 324A6FBC

The recoverable string S_r results from applying the mask generation function *MGF1* to the leftmost 856 (=1024–160–8) bits of S_i, and the leftmost 1 bit of S_r is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8).

S_r = 6F2BB975 71FE2EF2 05B66000 E9DD0665 6655C197 7F374E86 66D63655 6A5FEEEE
AF645555 B25F4556 7C4EE534 1F96FED8 6508C90A 9E3F11B2 6E8D4961 39ED3E55
ECE42860 A6FB3A08 17DAFBF1 3019D93E 1D382DA0 7264FE99 D9797D2F 0B777935
7CA7E74E E440D885 5B7DDF15 F000AC58 EE3FFF14 4845E771 907C0C83 324A6FBC

The recoverable integer f_r is the unsigned positive integer represented by S_r. Because the Jacobi symbol of f_r with respect to n equal to 1, the result is kept. f_r is raised to the power s modulo n. The result is represented by the temporary unsigned positive integer t.

t = A1B2C623 971FD3F2 83DE75E1 6A69EA07 262F2A0E DB41B5D5 E048DEB2 15DDCA0D
37A04E1E 87372B77 740CE252 3EF1EE7F F68D4781 819F797E DA8AD7AE F4D7EB7B
C7CE09AE 937FC765 452BABF2 56C9DE4D CEE7C866 C285AA58 09674464 BDD4708F
F37BC8F1 753C7E84 9D76EB1D 522F7FB9 6ABCC26F 591DF88A 33B736C2 82690912

Because the above result is greater than $n/2$, it is replaced by its complement to n. The binary string representing that integer as an unsigned positive integer is the signature $\Sigma = n - t$.

Σ = 1B38688C 96FCBAA7 15DE2022 8E8F339E E21F7CD8 EC1A1BB7 F084E029 0BFC5FE4
66D2C307 1679A619 3D851F34 69207CD8 36866425 17B8FCBC 0004A242 6DEF4C21
993BC91A B72318DC 6E57FB59 684E217E 456F9843 C8D313E3 F75DE73E FF313940
CADFDC12 7135A13F 68F64C8B 49C0F310 10EEEFFD 9D949DA8 D0867DB1 ED341863

The signed message consists of the 128 octets of the signature Σ together with the 6 octets of the non-recoverable message M_2, i.e., only 22 octets more than the message M.

E.2.3.3.2 The verification process

The signature Σ is a binary string representing an unsigned positive integer, which is less than $n/2$. That integer is squared modulo n, thus providing the resulting integer f_s.

f_s = 6F2BB975 71FE2EF2 05B66000 E9DD0665 6655C197 7F374E86 66D63655 6A5FEEEE
AF645555 B25F4556 7C4EE534 1F96FED8 6508C90A 9E3F11B2 6E8D4961 39ED3E55
ECE42860 A6FB3A08 17DAFBF1 3019D93E 1D382DA0 7264FE99 D9797D2F 0B777935
7CA7E74E E440D885 5B7DDF15 F000AC58 EE3FFF14 4845E771 907C0C83 324A6FBC

The verification process does not involve the Jacobi symbol. Because the least significant three bits of the resulting integer f_s are equal to '100', $f_r' = f_s$.

f_r' is represented as an unsigned positive integer by the recovered string S_r'. The mask generation function *MGF1* applied to the leftmost 856 (=1024–160–8) bits of S_r', thus providing the recovered intermediate string S_i'.

S_i' = 01616263 64626364 65636465 66646566 67656667 68666768 69676869 6A68696A
6B696A6B 6C6A6B6C 6D6B6C6D 6E6C6D6E 6F6D6E6F 706E6F70 716F7071 72707172
73717273 74727374 75737475 76747576 77757677 78767778 79777879 7A78797A
61797A61 627A6162 63616215 F000AC58 EE3FFF14 4845E771 907C0C83 324A6FBC

S_i' represents the recovered intermediate string processed as follows.

— The leftmost one bit of S_i' is set to '0' as $\delta = 1$ (δ = (1–1024) mod 8). The leftmost 7 bits of the resulting binary string are equal to '0'; it is followed by the border bit '1'; that 1 octet is removed on the left of S_i'.

— The rightmost octet of S_i' is equal to 'BC'; this octet is also removed on the right of S_i'.

Because the trailer is equal to 'BC', the hash function in use is implicitly known: RIPEMD-160 in this example.

The remaining string of 1008 bits is divided into two parts.

— M_1* consists of the leftmost 848 bits.

— H' consists of the rightmost 160 bits.

M_1* = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 6162

H' = 15F000AC 58EE3FFF 144845E7 71907C0C 83324A6F

Because the recovery is partial, the recovered message M* consists of the concatenation of M_1* and M_2*, the recovered and non-recoverable parts respectively.

M* = 61626364 62636465 63646566 64656667 65666768 66676869 6768696A 68696A6B
696A6B6C 6A6B6C6D 6B6C6D6E 6C6D6E6F 6D6E6F70 6E6F7071 6F707172 70717273
71727374 72737475 73747576 74757677 75767778 76777879 7778797A 78797A61
797A6162 7A616263 61626364 62636465

Another hash-code H'' is computed by applying dedicated hash-function 1 to the binary string of length 1072 (=64+848+160), that results from concatenating the 64 bits of the recovered message length C', the 848 bits of the recovered message part M_1* and the 160 bits of the hash-code of the non-recoverable message part $h(M_2^*)$. $H'' = h(C' \,||\, M_1^* \,||\, h(M_2^*))$.

H'' = 15F000AC 58EE3FFF 144845E7 71907C0C 83324A6F

Because the two hash-codes H' and H'' are identical, the signature Σ is accepted.

Bibliography

[1] M. Bellare and P. Rogaway, 'Random oracles are practical: a paradigm for designing efficient protocols'. In: *Proceedings of the first annual conference on Computer and Communications Security*, ACM, 1993, pp.62-73

[2] M. Bellare and P. Rogaway, 'Optimal asymmetric encryption – how to encrypt with RSA'. In: A. De Santis (editor), *Advances in Cryptology – Eurocrypt '94*, Lecture Notes in Computer Science **950** (1995), Springer-Verlag, pp.92-111

[3] M. Bellare and P. Rogaway, 'The exact security of digital signatures: How to sign with RSA and Rabin'. In: U.M. Maurer (editor), *Advances in Cryptology – Eurocrypt '96*, Lecture Notes in Computer Science **1070** (1996), Springer-Verlag, pp.399-416

[4] J.-S. Coron, 'On the exact security of full domain hashing'. In: M. Bellare (editor), *Advances in Cryptology – Crypto 2000*, Lecture Notes in Computer Science **1880** (2000), Springer-Verlag, pp.229-235

[5] J.-S. Coron, D. Naccache, and J.P. Stern, 'On the security of RSA padding'. In: M.J. Wiener (editor), *Advances in Cryptology – Crypto '99*, Lecture Notes in Computer Science **1666** (1999), Springer-Verlag, pp.1-18

[6] J.-S. Coron, D. Naccache, M. Tibouchi, and R.-P. Weinmann. 'Practical Cryptanalysis of ISO 9796-2 and Europay-Mastercard-Visa Signatures'. In: S. Halevi (editor), *Advances in Cryptology – Crypto 2009*, Lecture Notes in Computer Science **5677** (2009), Springer-Verlag, pp.428-444

[7] M. Girault and J.-F. Misarsky, 'Cryptanalysis of countermeasures proposed for repairing ISO 9796-1'. In: B. Preneel (editor), *Advances in Cryptology – Eurocrypt 2000*, Lecture Notes in Computer Science **1807** (2000), Springer-Verlag, pp.81-90

[8] F. Grieu, 'A chosen messages attack on the ISO/IEC 9796-1 signature scheme'. In: B. Preneel (editor), *Advances in Cryptology – Eurocrypt 2000*, Lecture Notes in Computer Science **1807** (2000), Springer-Verlag, pp.70-80

[9] IEEE Std 1363-2000, *Standard specifications for public key cryptography*

[10] IEEE Std 1363a-2004, *Standard specifications for public key cryptography — Amendment 1: Additional techniques*

[11] ISO/IEC 9796-3:2006, *Information technology — Security techniques — Digital signature schemes giving message recovery — Part 3: Discrete logarithm based mechanisms*

[12] ISO/IEC 9797-2:2002, *Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a dedicated hash-function*

[13] ISO/IEC 9798-1:1997[1)], *Information technology — Security techniques — Entity authentication — Part 1: General*

[14] ISO/IEC 14888 (all parts), *Information technology — Security techniques — Digital signatures with appendix*

1) ISO/IEC 9798-1:1997 has been cancelled and replaced by ISO/IEC 9798-1:2010.

[15] J. Jonsson, 'Security proofs for the RSA-PSS signature scheme and its variants'. *Proceedings of the 2nd NESSIE Workshop*, Royal Holloway, University of London, September 2001. Full version available in IACR cryptology archive **2001/053**

[16] B. Kaliski, 'On hash function firewalls in signature schemes'. In: B. Preneel (editor), *Cryptographers' Track RSA Conference 2002*, Lecture Notes in Computer Science **2271** (2002), Springer-Verlag, pp.1-16

ICS 35.040
Price based on 54 pages

This page deliberately left blank

British Standards Institution (BSI)

BSI is the national body responsible for preparing British Standards and other standards-related publications, information and services.

BSI is incorporated by Royal Charter. British Standards and other standardization products are published by BSI Standards Limited.

About us

We bring together business, industry, government, consumers, innovators and others to shape their combined experience and expertise into standards -based solutions.

The knowledge embodied in our standards has been carefully assembled in a dependable format and refined through our open consultation process. Organizations of all sizes and across all sectors choose standards to help them achieve their goals.

Information on standards

We can provide you with the knowledge that your organization needs to succeed. Find out more about British Standards by visiting our website at bsigroup.com/standards or contacting our Customer Services team or Knowledge Centre.

Buying standards

You can buy and download PDF versions of BSI publications, including British and adopted European and international standards, through our website at bsigroup.com/shop, where hard copies can also be purchased.

If you need international and foreign standards from other Standards Development Organizations, hard copies can be ordered from our Customer Services team.

Subscriptions

Our range of subscription services are designed to make using standards easier for you. For further information on our subscription products go to bsigroup.com/subscriptions.

With **British Standards Online (BSOL)** you'll have instant access to over 55,000 British and adopted European and international standards from your desktop. It's available 24/7 and is refreshed daily so you'll always be up to date.

You can keep in touch with standards developments and receive substantial discounts on the purchase price of standards, both in single copy and subscription format, by becoming a **BSI Subscribing Member**.

PLUS is an updating service exclusive to BSI Subscribing Members. You will automatically receive the latest hard copy of your standards when they're revised or replaced.

To find out more about becoming a BSI Subscribing Member and the benefits of membership, please visit bsigroup.com/shop.

With a **Multi-User Network Licence (MUNL)** you are able to host standards publications on your intranet. Licences can cover as few or as many users as you wish. With updates supplied as soon as they're available, you can be sure your documentation is current. For further information, email bsmusales@bsigroup.com.

BSI Group Headquarters

389 Chiswick High Road London W4 4AL UK

Revisions

Our British Standards and other publications are updated by amendment or revision.

We continually improve the quality of our products and services to benefit your business. If you find an inaccuracy or ambiguity within a British Standard or other BSI publication please inform the Knowledge Centre.

Copyright

Useful Contacts:

Customer Services
Tel: +44 845 086 9001
Email (orders): orders@bsigroup.com
Email (enquiries): cservices@bsigroup.com

Subscriptions
Tel: +44 845 086 9001
Email: subscriptions@bsigroup.com

Knowledge Centre
Tel: +44 20 8996 7004
Email: knowledgecentre@bsigroup.com

Copyright & Licensing
Tel: +44 20 8996 7070
Email: copyright@bsigroup.com

bsi.

...making excellence a habit.™