BRITISH STANDARD

BS ISO/IEC 9796-3:2006

Incorporating corrigendum September 2013

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

Part 3: Discrete logarithm based mechanisms

ICS 35.040

British Standards

National foreword

This British Standard is the UK implementation of ISO/IEC 9796-3:2006.

The ISO corrected text [15 September 2013] incorporates the following editorial corrections:

- The year of publication has been removed from references to ISO/IEC 15946-1.
- The last paragraph of 6.2.1 has been modified and ISO/IEC 15946-5 has been added to Clause 2.

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.

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 October 2006

ISBN 978 0 580 82245 2

Amendments/corrigenda issued since publication

Date	Comments
28 February 2014	Implementation of ISO corrected text 15 September 2013

INTERNATIONAL STANDARD

ISO/IEC 9796-3

Second edition
2006-09-15

Corrected version
2013-09-15

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

Part 3: Discrete logarithm based mechanisms

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

Partie 3: Mécanismes basés sur les logarithmes discrets

Reference number
ISO/IEC 9796-3:2006(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

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.

ISO/IEC 9796-3 was prepared by Joint Technical Committee ISO/IEC /JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 9796-3:2000), which has been technically revised. New mechanisms and object identifiers have been specified.

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*

This corrected version of ISO/IEC 9796-3:2006 incorporates the following corrections:

— The year of publication has been removed from references to ISO/IEC 15946-1.

— The last paragraph of 6.2.1 has been modified and ISO/IEC 15946-5 has been added to Clause 2.

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 only the public verification key and not the private signature key, it is computationally infeasible to produce a valid signature for any given message;

— the signatures produced by a signer can neither be used for producing a valid signature for any new message nor for recovering the signature key;

— it is computationally infeasible, even for the signer, to find two different messages with the same signature.

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 using the private signature key, called the ***signature generation process***;

— a process using the public verification key, called the ***signature verification process***.

There are two types of digital signature mechanisms:

— when, for each given private signature key, the signatures produced for the same message are the same, the mechanism is said to be ***non-randomized*** (or ***deterministic***) [see ISO/IEC 14888-1];

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

This part of ISO/IEC 9796 specifies randomized mechanisms.

Digital signature schemes 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 a part of it is recovered from the signature, the mechanism is named a ***signature mechanism giving message recovery***.

If the message is short enough, then the entire message can be included in the signature, and recovered from the signature in the signature verification process. Otherwise, a part of the message can be included in the signature and the rest of it is stored and/or transmitted along with the signature. The mechanisms specified in ISO/IEC 9796 give either total or partial recovery, aiming at reducing storage and transmission overhead.

This part of ISO/IEC 9796 includes six mechanisms, one of which was in ISO/IEC 9796-3:2000 and five of which are in ISO/IEC 15946-4:2004. The mechanisms specified in this part of ISO/IEC 9796 use a hash-function to hash the entire message. ISO/IEC 10118 specifies hash-functions. Some of the mechanisms specified in this part of ISO/IEC 9796 use a group on an elliptic curve over finite field. ISO/IEC 15946-1 describes the mathematical background and general techniques necessary for implementing cryptosystems based on elliptic curves defined over finite fields.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of patents concerning the mechanisms NR, ECMR and ECAO given in Clause 8, 10 and 11, respectively.

Area	**Patent no.**	**Issue date**	**Inventors**
NR [see Clause 8]	US 5 600 725, EP 0 639 907	1997-02-04	K. Nyberg and R. A. Rueppel
ECMR [see Clause 10]	JP H09-160492 (patent application)		A. Miyaji
ECAO [see Clause 11]	JP 3 434 251	2003-08-04	M. Abe and T. Okamoto

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from the following companies.

Patent no.	**Name of holder of patent right**	**Contact address**
US 5 600 725, EP 0 639 907	Certicom Corp.	5520 Explorer Drive, 4th Floor, Mississauga, Ontario, Canada L4W 5L1
JP H09-160492	Matsushita Electric Industrial Co., Ltd.	Matsushita IMP Building 19th Floor, 1-3-7, Siromi, Chuo-ku, Osaka 540-6319, Japan
JP 3 434 251	NTT Intellectual Property Center	9-11 Midori-Cho 3-chome, Musashino-shi, Tokyo 180-8585, Japan

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

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

NOTE 2 Any signature mechanism giving message recovery — for example, the mechanisms specified in this part of ISO/IEC 9796 — can be converted for provision of digital signatures with appendix. In this case, the signature is produced by application of the signature mechanism to a hash-token of the message.

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

Part 3: Discrete logarithm based mechanisms

1 Scope

This part of ISO/IEC 9796 specifies six digital signature schemes giving message recovery. The security of these schemes is based on the difficulty of the discrete logarithm problem, which is defined on a finite field or an elliptic curve over a finite field.

This part of ISO/IEC 9796 also defines an optional control field in the hash-token, which can provide added security to the signature.

This part of ISO/IEC 9796 specifies randomized mechanisms.

The mechanisms specified in this part of ISO/IEC 9796 give either total or partial message recovery.

NOTE For discrete logarithm based digital signature schemes with appendix, see ISO/IEC 14888-3.

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*

ISO/IEC 15946-1, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 1: General*

ISO/IEC 15946-5, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 5: Elliptic curve generation*

3 Terms and definitions

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

3.1
data input
octet string which depends on the entire message or a portion of the message and which forms a part of the input to the signature generation process

3.2
domain parameter
data item which is common to and known by or accessible to all entities within the domain

[ISO/IEC 14888-1:1998]

NOTE The set of domain parameters may contain data items such as hash-function identifier, length of the hash-token, maximum length of the recoverable part of the message, finite field parameters, elliptic curve parameters, or other parameters specifying the security policy in the domain.

3.3
elliptic curve
set of points $P = (x, y)$, where x and y are elements of an explicitly given finite field, that satisfy a cubic equation without any singular point, together with the "point at infinity" denoted by O

[ISO/IEC 15946-1:2002]

NOTE For a mathematical definition of an elliptic curve over an explicitly given finite field, see Clause A.4.

3.4
explicitly given finite field
set of all e-tuples over $[0, p - 1]$, where p is prime and $e \geq 1$, along with a "multiplication table"

NOTE 1 For a mathematical definition of an explicitly given finite field, see Clause A.3.

NOTE 2 For more detailed information on finite fields, see ISO/IEC 15946-1.

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

NOTE Adapted from ISO/IEC 10118-1:2000.

3.6
hash-function
function which maps strings of octets to fixed-length strings of octets, 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.

NOTE 1 Adapted from ISO/IEC 10118-1:2000.

NOTE 2 Computational feasibility depends on the specific security requirements and environment.

NOTE 3 For the purposes of this part of ISO/IEC 9796, the allowable hash-functions are those described in ISO/IEC 10118-2 and ISO/IEC 10118-3, with the following proviso:

— The hash-functions described in ISO/IEC 10118 map bit strings to bit strings, whereas in this part of ISO/IEC 9796, they map octet strings to octet strings. Therefore, a hash-function in ISO/IEC 10118-2 or ISO/IEC 10118-3 is allowed in this part of ISO/IEC 9796 only if the length in bits of the output is a multiple of 8, in which case the mapping between octet strings and bit strings is affected by the functions $\mathrm{OS2BSP}$ and $\mathrm{BS2OSP}$.

3.7
hash-token
concatenation of a hash-code and an optional control field which can be used to identify the hash-function and the padding method

[ISO/IEC 14888-1:1998]

NOTE The control field with the hash-function identifier is mandatory unless the hash-function is uniquely determined by the signature mechanism or by the domain parameters.

3.8
message
string of octets of any length

3.9
parameter generation process
process which gives as its output domain parameter and user keys

3.10
pre-signature
octet string computed in the signature generation process which is a function of the randomizer but which is independent of the message

NOTE Adapted from ISO/IEC 14888-1:1998.

3.11
private signature key
data item specific to an entity and usable only by this entity in the signature generation process

3.12
public verification key
data item which is mathematically related to a private signature key and is known by or accessible to all entities and which is used by the verifier in the signature verification process

3.13
randomized
dependent on a randomizer

[ISO/IEC 14888-1]

3.14
randomizer
secret integer produced by the signing entity in the pre-signature production process, and not predictable by other entities

NOTE Adapted from ISO/IEC 14888-1:1998.

3.15
signature
pair of an octet string and an integer for providing authentication, generated in the signature generation process

NOTE Adapted from ISO/IEC 14888-1:1998.

3.16
signature generation process
process which takes as inputs the message, the signature key and the domain parameters, and which gives as output the signature

NOTE Adapted from the definition of **signature process** in ISO/IEC 14888-1:1998.

3.17
signature verification process
process, which takes as its input the signed message, the verification key and the domain parameters, and which gives as its output the recovered message if valid

NOTE Adapted from the definition of **verification process** in ISO/IEC 14888-1:2008.

3.18
signed message
set of data items consisting of the signature, the part of the message which cannot be recovered from the signature, and an optional text field

[ISO/IEC 14888-1:1998]

3.19
user keys
data item of a set of private signature key and public verification key

4 Symbols, notation and conventions

4.1 Symbols and notation

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

A	entity, usually signer
B	entity, usually verifier
d	data input (octet string)
d'	recovered data input (octet string)
E	elliptic curve over explicitly given finite field
F	explicitly given finite field
G	generator of underlying group (finite field element / elliptic curve point)
h	(truncated) hash-token (octet string)
h'	recovered (truncated) hash-token (octet string)
h''	recomputed (truncated) hash-token (octet string)
Hash, Hash_1, Hash_2	hash-function
k	randomizer (integer)
KDF	key derivation function (synonym for MGF)
L_{clr}	length in octets of non-recoverable part (integer)
L_{dat}	length in octets of data input (integer)
L_F	length in octets of explicitly given finite field F (non-negative integer)
L_{rec}	(maximum) length in octets of recoverable part (integer)
L_{red}	length in octets of (added) redundancy (integer)
$L(x)$	length in octets of integer x or octet string x (non-negative integer)
L_{Hash}	length in octets of output of hash-function Hash (non-negative integer)
M	message (octet string)
M_{clr}	non-recoverable part of message (octet string)
M_{rec}	recoverable part of message (octet string)
M'	recovered message (octet string)
M'_{clr}	received non-recoverable part of message (octet string)
M'_{rec}	recovered part of message (octet string)
MGF	mask generation function
n	order of group generated by G (prime number)

O	point at infinity of elliptic curve
p	prime number
P	element dependent on the chosen key generation scheme, that is $P = G$ for Key Generation Scheme I and $P = Y_A$ for Key Generation Scheme II [see Clause 7.3]
Π	pre-signature (octet string)
Π'	recovered pre-signature (octet string)
q	prime power
Q	element dependent on the chosen key generation scheme, that is $Q = Y_A$ for Key Generation Scheme I and $Q = G$ for Key Generation Scheme II [see Clause 7.3]
r	first part of signature (octet string)
r'	first part of recovered signature (octet string)
s	second part of signature (integer)
s'	second part of recovered signature (integer)
x_A	private signature key of entity A
Y_A	public verification key of entity A
$\{0, 1\}^*$	set of finite bit strings
$\{0, 1\}^{8*}$	set of finite octet strings
$\{0, 1\}^{\ell}$	set of bit strings of length ℓ, where ℓ is a non-negative integer
$\{0, 1\}^{8\ell}$	set of octet strings of length ℓ, where ℓ is a non-negative integer
$[a, b]$	set of integers x satisfying $a \le x \le b$, where a and b are integers
$\|x\|$	length of bit string x
$\|X\|$	cardinality of set X
$[x]^{\ell}$	leftmost ℓ-bits of octet string x, appending zeros to the right when $8\ell > L(x)$
$[x]_{\ell}$	rightmost ℓ-bits of octet string x, appending zeros to the left when $8\ell > L(x)$
$x \bmod n$	$r \in [0, n-1]$ such that $(x - r)$ is divisible by n, where x is an integer
$x \oplus y$	bitwise exclusive-OR operation of bit strings x and y
$x \parallel y$	concatenation of bit strings x and y
$X \times Y$	Cartesian product of sets X and Y

4.2 Conversion functions and mask generation functions

For the purposes of this document, the following conversion functions and mask generation functions are used.

BS2IP	bit-string-to-integer primitive [see Clause B.2]
BS2OSP	bit-string-to-octet-string primitive [see Clause B.1]
EC2OSP	elliptic-curve-to-octet-string primitive [see Clause B.6]
FE2IP	finite-field-element-to-integer primitive [see Clause B.4]
FE2OSP	finite-field-element-to-octet-string primitive [see Clause B.5]
I2BSP	integer-to-bit-string primitive [see Clause B.2]
I2OSP	integer-to-octet-string primitive [see Clause B.3]
MGF1	mask generation function 1 [see Clause C.2]
MGF2	mask generation function 2 [see Clause C.3]
OS2BSP	octet-string-to-bit-string primitive [see Clause B.1]
OS2ECP	octet-string-to-elliptic-curve primitive [see Clause B.6]
OS2FEP	octet-string-to-finite-field-element primitive [see Clause B.5]
OS2IP	octet-string-to-integer primitive [see Clause B.3]

4.3 Legend for figures

The following legend is used for the figures in Clause 7 depicting the signature generation and verification processes for digital signatures giving message recovery.

STEP OF THE PROCESS — step of the process


mandatory data flow


optional data flow

5 Binding between signature mechanisms and hash-functions

Use of the signature schemes specified in this part of ISO/IEC 9796 requires the selection of a hash-function Hash. ISO/IEC 10118 specifies hash-functions. 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.

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.

NOTE 1 One of the security requirements for the hash-function Hash used in this part of ISO/IEC 9796 is so-called "collision-resistance."

NOTE 2 There are various ways to accomplish this binding. The following options are listed in order of increasing risk:

a) 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 (otherwise known as SHA-1) from ISO/IEC 10118-3;

b) 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; and

c) 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 3 The "other method" referred to in paragraph c) immediately above could be in the form of a hash-function identifier included in the octet string representative d. If the hash-function identifier is included in d in this way then an attacker cannot fraudulently reuse an existing signature with the same octet string d_1 and a different d_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, in this latter case and using the weak hash-function, an attacker can still find a new signature with a "random" d_1.

NOTE 4 The attack mentioned in Note 3 that yields a new signature with a "random" d_1 can be prevented by requiring the presence of a specific structure in d_1. For instance, one may impose a length limit on d_1 that is sufficiently less than the capacity of the signature scheme. For some digital signature schemes, a length limit on d_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 MGF 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.

6 Framework for digital signatures giving message recovery

6.1 Processes

Clauses 6.2 through 6.4 contain a high-level description of a general model for the six signature schemes specified in this part of ISO/IEC 9796. A detailed description of the general model is provided in Clause 7.

A digital signature scheme specified in this part of ISO/IEC 9796 is defined by the specification of the following processes:

— parameter generation process;

— signature generation process;

— signature verification process.

6.2 Parameter generation process

6.2.1 Domain parameters

The parameters can be divided into domain parameters and user keys. The domain parameters consist of parameters to define a finite group, such as a multiplicative group of a finite field or an additive group on an elliptic curve over a finite field, and other public information which is common to and known by or accessible to all entities within the domain. As well as the domain parameters specific to the cryptographic scheme in use, the following parameters must be specified:

— an identifier for the digital signature scheme used;

— the type of redundancy;

— (optional) a hash function Hash;

— the user key generation procedures.

For the implementation techniques and the mathematical background for an additive group on an elliptic curve over a finite field, ISO/IEC 15946-1 shall be referred. For the methods to construct an elliptic curve over a finite field, ISO/IEC 15946-5 shall be referred.

6.2.2 User keys

Each entity has its own public and private keys. The user keys of entity A consist of the following:

— the private signature key x_A;

— the public verification key Y_A;

— (optional) other information, which is specific to the entity A, for the use in the signature generation and/or verification process.

NOTE 1 User keys are valid only within the context of a specified set of domain parameters.

NOTE 2 The signature verifier may require assurance that the domain parameters and public verification key are valid, otherwise there is no assurance of meeting the intended security even if the signature verifies. The signer may also require assurance that the domain parameters and public verification key are valid, otherwise an adversary may be able to generate signatures that verify.

6.3 Signature generation process

The following data items are required for the signature generation process:

— the domain parameters;

— the signer A's private signature key x_A;

— a message M.

For all the schemes specified in this part of ISO/IEC 9796, the signature generation process consists of the following procedures:

a) splitting the message;

b) (optional) computation of redundancy, or computation of the message digest;

c) computations in a finite group, which is either the multiplicative group of a finite field or the additive group on an elliptic curve over a finite field;

d) computations modulo the group order of the base element G;

e) formatting the signed message.

The output of the signature generation process is a pair (r, s) that constitutes A's digital signature of the message M.

6.4 Signature verification process

The following data items are required for the signature verification process:

— the domain parameters;

— the signer A's public verification key Y_A;

— the non-recoverable part of the message M'_{clr} (if any);

— the received signature for M, represented as an octet string r' and an integer s'.

For all the schemes the signature verification process consists of some or all of the following procedures:

a) signature size verification;

b) computations in a finite group, which is either the multiplicative group of a finite field or the additive group on an elliptic curve over a finite field;

c) computations modulo the group order of the base element G;

d) recovering the data input or the message;

e) signature checking.

If all procedures are passed successfully, the signature is accepted by the verifier; otherwise it is rejected.

7 General model for digital signatures giving message recovery

7.1 Requirements

7.1.1 Domain parameters

Users who wish to employ one of the digital signature mechanisms specified in this part of ISO/IEC 9796 shall select the following domain parameters of the digital signature scheme:

a) an explicitly given finite field F, or an elliptic curve E over an explicitly given finite field F;

b) an element G in F or E of prime order n.

Agreement on these choices amongst the users is essential for the purpose of the operation of the digital signature mechanism giving message recovery.

NOTE 1 The size of n affects the level of security offered by the scheme and shall be chosen to meet the defined security objectives.

NOTE 2 The two possible groups with which this scheme may be used are normally written using multiplicative notation (for the multiplicative group of the finite field) and additive notation (for the group of points on an elliptic curve). In Clause 7, the multiplicative notation is used, in order to simplify the presentation.

NOTE 3 For the definition of an explicitly given finite field, see Clause A.3.

NOTE 4 For the definition of an elliptic curve over an explicitly given finite field, see Clause A.4.

NOTE 5 For efficient implementations and cryptographic techniques related to the groups on elliptic curves, see ISO/IEC 15946-1.

7.1.2 Type of redundancy

Users shall select the type of redundancy, which shall be

- natural redundancy,
- added redundancy, or
- both.

Agreement on the type of redundancy amongst the users is essential for the purpose of the operation of the digital signature mechanism giving message recovery.

If users use added redundancy, the length in octets of added redundancy, L_{red}, shall be fixed. A message with added redundancy may be constructed by the hash token of the message or of the recoverable message.

If users use natural redundancy alone, then L_{red} is set equal to 0. A message with natural redundancy means that the message includes redundancy naturally, such as the use of ASCII characters, or that the redundancy of the message is verifiable implicitly in some applications.

The natural or added redundancy may be anything agreed upon as long as it can be checked by the communicating parties. Total redundancy, which consists of natural redundancy and added redundancy, shall be greater than some minimum value specified by the application. In general natural redundancy alone shall only be used for total message recovery.

NOTE The value of the parameter L_{red} also affects the security level of the signatures giving message recovery.

7.2 Summary of functions and procedures

The signature schemes specified in this part of ISO/IEC 9796 give message recovery. More precisely, some of the data which is input to the signature generation function is recovered from the signature as part of the signature verification procedure.

The signature scheme consists of the following functions and procedures:

- user key generation process;
- signature generation process;
- signature verification process.

7.3 User key generation process

One of the following two methods shall be used to compute the key pair consisting of the public verification and the private signature key (the signing entity shall keep the private signature key secret):

a) Key generation I

Given a valid set of domain parameters, a private signature key and corresponding public verification key may be generated as follows:

1) Select a random or pseudorandom integer x_A in the set $[1, n-1]$. The integer x_A must be protected from unauthorised disclosure and be unpredictable;

2) Compute the element $Y_A = G^{x_A}$;

3) The key pair is (Y_A, x_A), where Y_A will be used as public verification key, and x_A is the private signature key.

To allow an unified representation of the algorithms, put $P = G$ and $Q = Y_A$.

b) Key generation II

Given a valid set of domain parameters, a private signature key and corresponding public verification key may be generated as follows:

1) Select a random or pseudorandom integer e in the set $[1, n-1]$ and compute an integer x_A in the interval $[1, n-1]$ with the property $x_A e = 1 \bmod n$. The integer x_A must be protected from unauthorised disclosure and be unpredictable;

2) Compute the element $Y_A = G^e$, and then erase the integer e in a secure manner;

3) The key pair is (Y_A, x_A), where Y_A will be used as public verification key, and x_A is the private signature key.

To allow an unified representation of the algorithms, put $P = Y_A$ and $Q = G$.

Prior to use of the public verification key the verifier shall have assurance about its validity and ownership. This validation may be obtained by various means, see Clause 6.2.2.

NOTE 1 Some schemes use the range $[1, n-2]$ for the private signature key x_A.

NOTE 2 Key generation I is the more popular method and is often used. In some environments where modular inversion is expensive, Key generation II might be useful.

7.4 Signature generation process

7.4.1 Procedures

Figure 1 shows the signature generation process, which consists of the following procedures:

a) producing a randomizer and the pre-signature;

b) splitting the message;

c) producing the data input;

d) computing the signature;

e) formatting the signed message.

NOTE Each mechanism may require scheme-dependent domain parameters other than those shown in Figure 1.

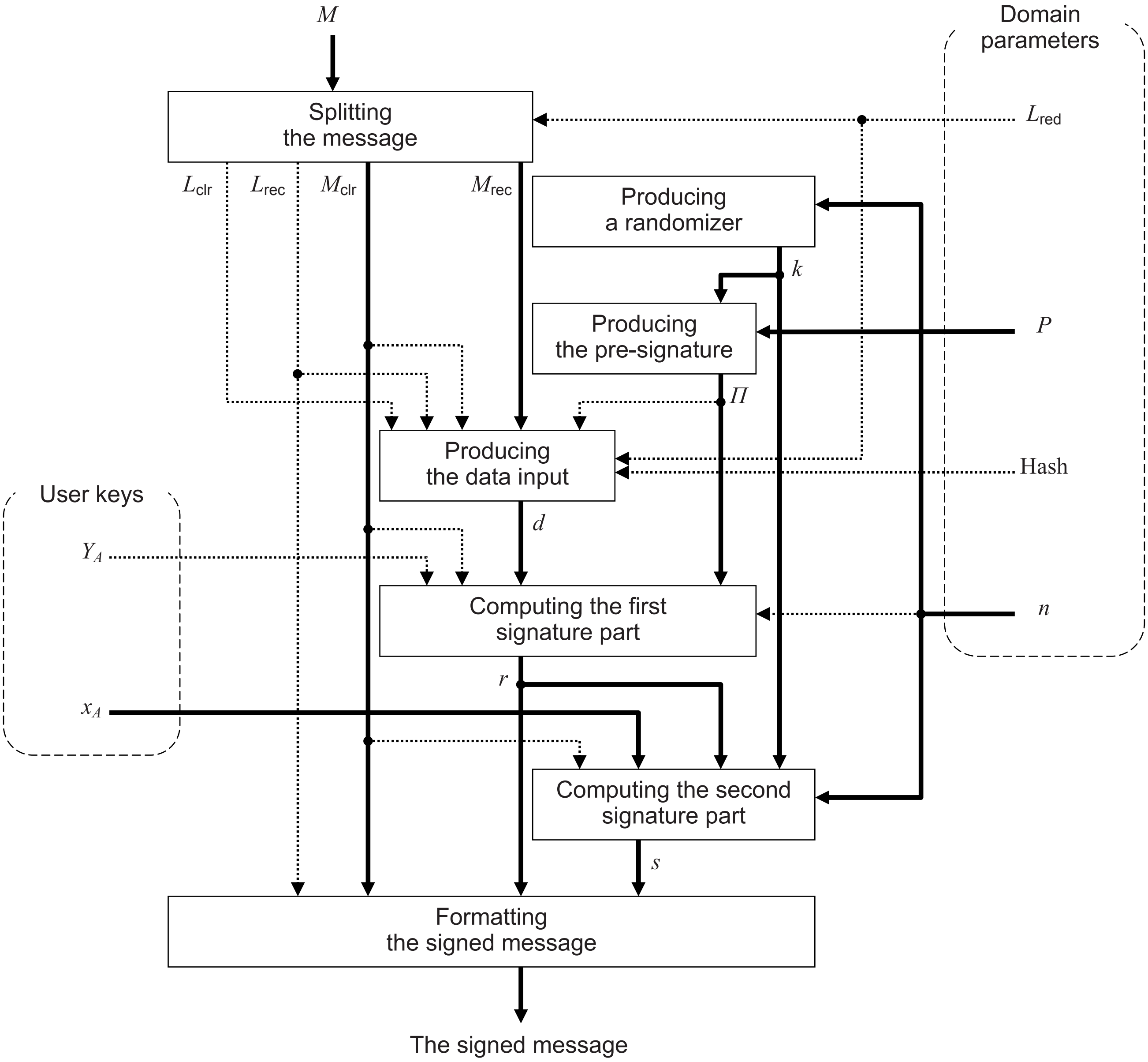

Figure 1 — The signature generation process

7.4.2 Producing a randomizer and the pre-signature

Prior to each signature computation the signing entity must have a fresh, secret randomizer value available. The randomizer is an integer k such that $1 \le k \le n-1$. The implementation of the signature scheme must ensure that the following two requirements are satisfied:

- randomizer generation shall be executed in such a way that the probability that the same randomizer is used to produce signatures for two different messages shall be negligible;

- used randomizer values shall never be disclosed; once used, they shall be destroyed.

First a randomizer k, which is an integer, is produced. Then the pre-signature Π, which is an octet string, is computed as a function of the randomizer. The pre-signature is an intermediate data item that is produced during the signature generation process in any randomized signature mechanism. The pre-signature is a public data item, while the value of the randomizer shall be available only to the signature generation process.

NOTE 1 Disclosure of a randomizer after use may jeopardise the secrecy of the private key. Used randomizers are never required again by the signer or verifier and should be securely erased. If the same value of the randomizer is used to produce signatures for two different messages, or if the randomizer for a signature is disclosed, then it might be possible to recover the private key from the signatures.

NOTE 2 Randomizers may be produced and corresponding pre-signatures may be computed offline. In this case, the randomizers should be stored securely for future use by the signature generation process.

7.4.3 Splitting the message

The message M is split into the recoverable part M_{rec} and the non-recoverable part M_{clr} of the message, and L_{rec} and L_{clr} are defined to be the length in octets of the recoverable part M_{rec} and the non-recoverable part M_{clr}, respectively.

7.4.4 Producing the data input

The input to the data input function is the recoverable part of the message M_{rec} with added redundancy, or the recoverable part of the message M_{rec} with natural redundancy. The inputs may optionally include the non-recoverable part M_{clr} , the lengths L_{rec} and L_{clr}. If added redundancy is used, the data input involves producing the hash-token. The hash-token is formed by the hash-code itself, or with the hash-function identifier concatenated to the right of the hash-code, where the hash-code is computed by hashing the (recoverable part of) message. The choice of whether or not the hash-token includes the hash-function identifier shall be controlled by the domain parameters. The output of the data input function is d, which is an octet string.

NOTE 1 The choice of data input may be determined by each application or signature scheme.

NOTE 2 See Annex D for an example method of producing the data input with added redundancy.

7.4.5 Computing the signature

The signatures produced by the schemes in this part of ISO/IEC 9796 have two parts r and s. The first part r is an octet string which is computed as a function of the pre-signature Π and the data input d (and optionally other parameters), where d is an octet string that depends upon the message. The second part s is an integer such that $0 < s < n$ and computed as a function of the first part r, the randomizer k, and the private signature key x_A (and optionally other parameters).

7.4.6 Formatting the signed message

Knowledge of the length of the recoverable part of the message is necessary for the successful opening and verification of the signed message. This information must be given by the domain parameter, included in the signed message and/or retrieved from the data input d.

The signed message consists of the following data items:

— the non-recoverable part M_{clr} of the message;

— the first part r of the signature;

— the second part s of the signature;

— (optional) the length L_{rec} of the recoverable part of the message.

7.5 Signature verification process

7.5.1 Procedures

Figure 2 shows the signature verification process, which consists of the following procedures:

a) opening the signed message;

b) signature size verification;

c) recovering the pre-signature or the data input;

d) recovering the data input or the message;

e) re-computing the hash-token(optional);

f) checking the signature.

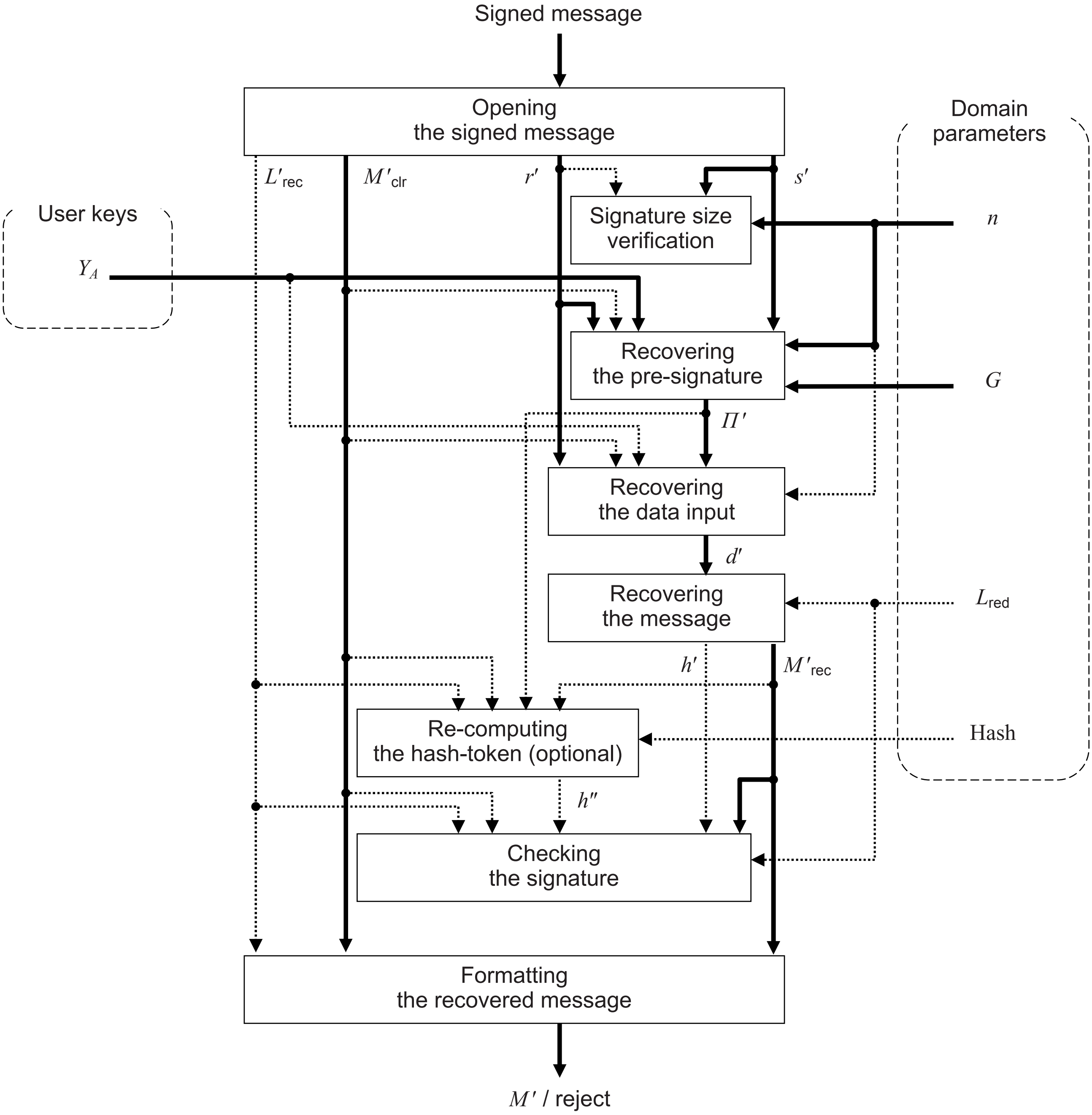

Figure 2 — Signature verification process

Checking the signature consists of

— comparing the recovered and recomputed (truncated) hash-tokens, or

— verifying the redundancy.

7.5.2 Opening the signed message

When starting this step, the verifier must have the following information available:

- the lengths of the different signature/message parts included in the signed message;
- the value of the parameter L_{red}.

The verifier extracts the following different parts of the signed message:

- the non-recoverable part of the message;
- the first part r' of the signature;
- the second part s' of the signature;
- (optional) the length L'_{rec} of the recoverable message part.

7.5.3 Signature size verification

The verifier shall verify the size of the parts of a signature.

7.5.4 Recovering the pre-signature

At the beginning of this step the verifier must have the following information available:

- the public parameters which specify the signature scheme in use;
- the public verification key Y_A of the signing entity.

The computations in this step are specific to the signature scheme in use. The pre-signature is determined by the public verification key Y_A. Given the signature (r', s'), the pre-signature Π' is recovered.

7.5.5 Recovering the data input or the message

Given the first part r' of the signature and the recovered pre-signature Π', the data input d' is recovered. The recovered data input d' is an octet string.

7.5.6 Re-computing the hash-token (optional)

First, the hash-function used by the signing entity in Clause 7.4 is identified, possibly by the domain parameter and/or by retrieving the hash-function identifier from the recovered hash-token. Then the hash-code is recomputed by hashing the message.

The recomputed hash-code is used to obtain the recomputed hash-token by optionally concatenating the hash-function identifier.

7.5.7 Checking the signature

Checking the signature consists of

- comparing the recomputed (truncated) hash-token h'' with the recovered (truncated) hash-token h', or
- verifying the added, and/or natural redundancy of the recovered message.

8 NR (Nyberg-Rueppel message recovery signature)[1]

8.1 Domain parameter and user keys

The domain parameter specifies a multiplicative group of an explicitly given finite field F.

The length of the data input d in octets, L_{dat}, is set equal to a fixed value less than or equal to $L(n) - 1$.

The keys of the NR signature scheme are produced as follows:

a) A's private signature key x_A which is a random integer in the interval $[1, n - 1]$;

b) A's public verification key Y_A computed as in Clause 7.3.

NOTE For the definition of an explicitly given finite field, see Clause A.3.

8.2 Signature generation process

8.2.1 Input and output

The input to the signature generation process consists of

— the domain parameters,

— the private signature key x_A, and

— a message M to be signed.

The output of the signature generation process is a pair $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n - 1]$ that constitutes A's digital signature to the message M.

8.2.2 Producing a randomizer and the pre-signature (finite field computations)

The pre-signature $\Pi \in \{0, 1\}^{8*}$ shall be computed by the following or an equivalent sequence of steps:

a) Select a random integer k in the interval $[1, n - 1]$;

b) Compute the finite field element $R = P^k$;

c) Convert R to an octet string $\Pi = \mathrm{FE2OSP}_F(R)$.

8.2.3 Producing the data input

The data input $d \in \{0, 1\}^{8L\mathrm{dat}}$ is produced from the message M; see Clauses 7.4.2 and 7.4.3.

1) This signature mechanism is based on a scheme defined in [9].

8.2.4 Computing the signature (arithmetic operations modulo *n*)

The signature $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n-1]$ shall be computed by the following or an equivalent sequence of steps:

a) Convert d to an integer $\delta = \mathrm{OS2IP}(d)$; note that $\delta \in [0, n-1]$;

b) Compute $\pi = \mathrm{OS2IP}(\Pi) \bmod n$;

c) Compute $\tilde{r} = (\delta + \pi) \bmod n$;

d) Compute $s = (k - x_A \tilde{r}) \bmod n$;

e) Convert $r = \mathrm{I2OSP}(\tilde{r}, L(n))$;

f) Erase k.

If the signature generation process yields either $\tilde{r} = 0$ or $s = 0$, then the process of signature generation must be repeated with a new random value k.

8.2.5 Formatting the signed message

The pair $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n-1]$ constitutes A's signature on the message M.

8.3 Signature verification process

8.3.1 Input and output

The signature verification process consists of three steps: calculation of the message digest, finite field computations, and signature checking.

The input to the signature verification process consists of

— the domain parameters,

— A's public verification key Y_A,

— the received signature for M, represented as an octet string r' and an integer s', and

— the non-recoverable message M'_{clr} (if any).

The output of the signature verification process is either the recovered data input d' or "reject."

8.3.2 Signature size verification

Verify that and $r' \in \{0, 1\}^{8L(n)}$, $0 < \mathrm{OS2IP}(r') < n$ and $0 < s' < n$; if not, then reject the signature.

8.3.3 Recovering the pre-signature (finite field computations)

The pre-signature shall be recovered from the received signature (r', s') by the following or an equivalent sequence of steps:

a) Convert $\tilde{r}' = \mathrm{OS2IP}(r')$;

b) Compute $R' = P^{s'} Q^{\tilde{r}'}$;

c) Convert R' to an octet string $\Pi' = \mathrm{FE2OSP}_F(R')$.

8.3.4 Recovering the data input or the message

The data input shall be recovered from the first part of the received signature r' and the recovered pre-signature Π' by the following or an equivalent sequence of steps:

a) Compute $\pi' = \mathrm{OS2IP}(\Pi') \bmod n$;

b) Compute $\delta' = (\vec{r}' - \pi') \bmod n$;

c) Convert δ' to an octet string $d' = \mathrm{I2OSP}(\delta', L_{\mathrm{dat}})$.

8.3.5 Checking the signature

Check the redundancy. If it is correct, output d', otherwise reject.

9 ECNR (Elliptic Curve Nyberg-Rueppel message recovery signature)

9.1 Domain parameter and user keys

The domain parameter specifies an additive group of order n in an elliptic curve E over an explicitly given finite field.

The length of the data input d in octets, L_{dat}, is set equal to a fixed value less than or equal to $L(n) - 1$.

The keys of the ECNR signature scheme are produced as follows:

a) A's private signature key x_A which is a random integer in the interval $[1, n - 1]$;

b) A's public verification key Y_A computed as in Clause 7.3.

NOTE 1 For the definition of an explicitly given finite field, see Clause A.3.

NOTE 2 For the definition of an elliptic curve over an explicitly given finite field, see Clause A.4.

9.2 Signature generation process

9.2.1 Input and output

The input to the signature generation process consists of

— the domain parameters,

— the private signature key x_A, and

— a message M to be signed.

The output of the signature generation process is a pair $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n - 1]$ that constitutes A's digital signature to the message M.

9.2.2 Producing a randomizer and the pre-signature (elliptic curve computations)

The pre-signature $\Pi \in \{0, 1\}^{8(L_F+1)}$ shall be computed by the following or an equivalent sequence of steps:

a) Select a random integer k in the interval $[1, n-1]$;

b) Compute the elliptic curve point $R = kP$;

c) Convert R to an octet string $\Pi = \mathrm{EC2OSP}_E(R, \texttt{compressed})$.

NOTE For the definition of the conversion function EC2OSP with the format specifier `compressed`, see Clause B.6.

9.2.3 Producing the data input

The data input $d \in \{0, 1\}^{8L_{\mathrm{dat}}}$ is produced from the message M; see Clauses 7.4.2 and 7.4.3.

9.2.4 Computing the signature (arithmetic operations modulo n)

The signature $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n-1]$ shall be computed by the following or an equivalent sequence of steps:

a) Convert d to an integer $\delta = \mathrm{OS2IP}(d)$; note that $\delta \in [0, n-1]$;

b) Compute $\pi = \mathrm{OS2IP}(\Pi) \bmod n$;

c) Compute $\tilde{r} = (\delta + \pi) \bmod n$;

d) Compute $s = (k - x_A \tilde{r}) \bmod n$;

e) Convert $r = \mathrm{I2OSP}(\tilde{r}, L(n))$;

f) Erase k.

If the signature generation process yields either $\tilde{r} = 0$ or $s = 0$, then the process of signature generation must be repeated with a new random value k.

9.2.5 Formatting the signed message

The pair $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n-1]$ constitutes A's signature on the message M.

9.3 Signature verification process

9.3.1 Input and output

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of

— the domain parameters,

— A's public verification key Y_A,

— the received signature for M, represented as an octet string r' and an integer s', and

— the non-recoverable message M'_{clr} (if any).

The output of the signature verification process is either the recovered data input d' or "reject."

9.3.2 Signature size verification

Verify that $\mathrm{OS2IP}(r') \neq 0 \bmod n$ and $r' \in \{\mathtt{0},\mathtt{1}\}^{8L(n)}$ and $0 < s' < n$; if not, then reject the signature.

9.3.3 Recovering the pre-signature (elliptic curve computations)

The pre-signature shall be recovered from the received signature (r', s') by the following or an equivalent sequence of steps:

a) Convert $\tilde{r}' = \mathrm{OS2IP}(r')$;

b) Compute $R' = s'P + \tilde{r}'Q$;

c) Convert R' to an octet string $\Pi' = \mathrm{EC2OSP}_E(R', \mathtt{compressed})$.

9.3.4 Recovering the data input or the message

The data input shall be recovered from the first part of the received signature r' and the recovered pre-signature Π' by the following or an equivalent sequence of steps:

a) Compute $\pi' = \mathrm{OS2IP}(\Pi') \bmod n$;

b) Compute $\delta' = (r' - \pi') \bmod n$;

c) Convert δ' to an octet string $d' = \mathrm{I2OSP}(\delta', L_{\mathrm{dat}})$.

9.3.5 Checking the signature

Check the redundancy. If it is correct, output d', otherwise reject.

10 ECMR (Elliptic Curve Miyaji message recovery signature)[2)]

10.1 Domain parameter and user keys

The domain parameter specifies an additive group on an elliptic curve as a finite group. The keys of the ECMR signature scheme are produced as follows:

a) A's private signature key x_A which is a random integer in the interval $[1, n-1]$;

b) A's public verification key Y_A computed as in Clause 7.3.

A also selects a function:

— $\mathrm{Mask} : \{\mathtt{0},\mathtt{1}\}^{8*} \rightarrow \{\mathtt{0},\mathtt{1}\}^{8L(n)}$, such that $\mathrm{Mask}(x) = [\mathrm{Hash}(x)]_{8L(n)}$, $\mathrm{MGF1}(x, L(n))$ or $\mathrm{MGF2}(x, L(n))$, where $\mathrm{Hash} : \{\mathtt{0},\mathtt{1}\}^{8*} \rightarrow \{\mathtt{0},\mathtt{1}\}^{8L_{\mathrm{Hash}}}$.

NOTE 1 For the definition of an explicitly given finite field, see Clause A.3.

NOTE 2 For the definition of an elliptic curve over an explicitly given finite field, see Clause A.4.

2) This signature mechanism is based on a scheme defined in [8].

10.2 Signature generation process

10.2.1 Input and output

The input to the signature generation process consists of

— the domain parameters,

— the private signature key x_A, and

— the data d with added or natural redundancy in $\{\mathtt{0}, \mathtt{1}\}^{8L(n)}$.

The data d is produced from the message, see Clauses 7.4.2 and 7.4.3. The output of the signature generation process is a pair $(r, s) \in \{\mathtt{0}, \mathtt{1}\}^{8L(n)} \times [1, n-1]$ that constitutes A's digital signature to the data.

10.2.2 Producing a randomizer and the pre-signature (elliptic curve computations)

The pre-signature $\Pi \in \{\mathtt{0}, \mathtt{1}\}^{8(2LF+1)}$ shall be computed by the following or an equivalent sequence of steps:

a) Select a random integer k in the interval $[1, n-1]$;

b) Compute the elliptic curve point $R = kP$;

c) Compute $\Pi = \mathrm{Mask}(\mathrm{EC2OSP}_E(R, \mathtt{uncompressed}))$.

NOTE For the definition of the conversion function EC2OSP with the format specifier `uncompressed`, see Clause B.6.

10.2.3 Computing the signature (computations modulo *n*)

The signature $(r, s) \in \{\mathtt{0}, \mathtt{1}\}^{8L(n)} \times [1, n-1]$ shall be computed by the following or an equivalent sequence of steps:

a) Compute $r = d \oplus \Pi$;

b) Compute $s = (\mathrm{OS2IP}(r)k - \mathrm{OS2IP}(r) - 1) / (x_A + 1) \bmod n$;

c) Erase k.

If the signature generation process yields either $s = 0$ or $\mathrm{OS2IP}(r) \bmod n = 0$, then the process of signature generation must be repeated with a new random value k.

10.2.4 Formatting the signed message

The pair $(r, s) \in \{\mathtt{0}, \mathtt{1}\}^{8L(n)} \times [1, n-1]$ constitutes A's signature on the data d.

10.3 Signature verification process

10.3.1 Input and output

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of

— the domain parameters,

— A's public verification key Y_A,

— the received signature for d, represented as an octet string r' and an integer s',

— the function Mask, and

— the non-recoverable message M'_{clr} (if any).

The output of the signature verification process is either the recovered data d' or "reject."

10.3.2 Signature size verification

Verify that $\mathrm{OS2IP}(r') \neq 0 \bmod n$ and $r' \in \{\mathtt{0}, \mathtt{1}\}^{8L(n)}$ and $0 < s' < n$; if not, then reject the signature.

10.3.3 Recovering the pre-signature (elliptic curve computations)

The pre-signature shall be recovered from the received signature (r', s') by the following or an equivalent sequence of steps:

a) Compute $R' = ((1 + \mathrm{OS2IP}(r') + s') / \mathrm{OS2IP}(r'))P + (s' / \mathrm{OS2IP}(r'))Q$;

b) Compute $\Pi' = \mathrm{Mask}(\mathrm{EC2OSP}_E(R', \mathtt{uncompressed}))$.

10.3.4 Recovering the data input or the message

Compute $d' = r' \oplus \Pi'$.

10.3.5 Checking the signature

Check the redundancy. If it is correct, output d', otherwise reject.

11 ECAO (Elliptic Curve Abe-Okamoto message recovery signature)[3)]

11.1 Domain parameter

The domain parameter specifies an additive group of order n, with a base element G, in an elliptic curve E over an explicitly given finite field F.

3) This signature mechanism is based on a scheme defined in [2].

The length of added redundancy, L_{red}, corresponds to the security parameter and shall be chosen to achieve security objectives.

In addition, A uses two hash functions and a mask generation function

— $\mathrm{Hash}_1 : \{0, 1\}^{8*} \rightarrow \{0, 1\}^{8L_{\mathrm{red}}}$,

— $\mathrm{Hash}_2 : \{0, 1\}^{8*} \rightarrow \{0, 1\}^{8(L_F + 1 - L_{\mathrm{red}})}$, and

— $\mathrm{MGF} : \{0, 1\}^{8*} \rightarrow \{0, 1\}^{8(L(n) + K)}$.

Here K is a non-negative integer that corresponds to the security parameter. The function MGF is defined as $\mathrm{MGF}(x) = \mathrm{MGF1}(x, L(n) + K)$ for $x \in \{0, 1\}^{8*}$.

NOTE 1 For the definition of an explicitly given finite field, see Clause A.3.

NOTE 2 For the definition of an elliptic curve over an explicitly given finite field, see Clause A.4.

NOTE 3 Since the non-recoverable message part is processed with computing the second part of the signature (and indeed only the recoverable message part is involved in computing the added redundancy), normally $L_{\mathrm{red}} = \lfloor L(n) / 2 \rfloor$ is used in ECAO for both total and partial message recoveries; see Clauses 11.3.3 and 11.3.4.

NOTE 4 Since the non-recoverable message part is input to MGF and the output of MGF is taken $\bmod n$, a larger value of K achieves a higher security level. The value $K = L(n)$ is recommended for use in ECAO; see Clause 11.3.4.

11.2 User keys

The keys of the ECAO signature scheme are produced as follows:

a) A's private signature key x_A which is a random integer in the interval $[1, n - 1]$;

b) A's public verification key Y_A computed as in Clause 7.3.

The base element G and the public verification key Y_A together provide the public data item (P, Q); the knowledge of which key generation scheme is used is public information and must be provided either as a domain parameter or along with the public verification key Y_A; see Clause 7.3.

11.3 Signature generation process

11.3.1 Input and output

The input to the signature generation process consists of

— the domain parameters,

— the private signature key x_A, and

— a message M to be signed.

The output of the signature generation process is a pair $(r, s) \in \{0, 1\}^{8(L_F + 1)} \times [1, n - 1]$ that constitutes A's digital signature to the message M. The signature (r, s) together with the non-recoverable message part M_{clr} constitutes the signed message.

11.3.2 Producing a randomizer and the pre-signature (elliptic curve computations)

The pre-signature $\Pi \in \{\mathtt{0, 1}\}^{8(L_F + 1)}$ shall be computed by the following or an equivalent sequence of steps:

a) Select a random integer k in the interval $[1, n - 1]$;

b) Compute the elliptic curve point $R = kP$;

c) Convert R to an octet string $\Pi = \mathrm{EC2OSP}_E(R, \mathtt{compressed})$.

NOTE For the definition of the conversion function EC2OSP with the format specifier `compressed`, see Clause B.6.

11.3.3 Splitting the message and producing the data Input

The maximum length of the recoverable part, $L_{\max}$, is set equal to $L_F - L_{\mathrm{red}}$. Split the message M into the recoverable part M_{rec} and the non-recoverable part M_{clr} so that the following two conditions are satisfied:

— $M = M_{\mathrm{rec}} \| M_{\mathrm{clr}}$;

— $L(M_{\mathrm{rec}}) \leq L_{\max}$.

Note that resulting octet strings M_{rec} or M_{clr} might be null.

Then form an octet string $\tilde{M}_{\mathrm{rec}}$ by the following or an equivalent sequence of steps:

a) Compute $\mathrm{pad} = \mathrm{I2OSP}(1, L_{\max} + 1 - L(M_{\mathrm{rec}}))$;

b) Compute $\tilde{M}_{\mathrm{rec}} = \mathrm{pad} \| M_{\mathrm{rec}}$.

Now the data input $d \in \{\mathtt{0, 1}\}^{8(L_F + 1)}$ is computed from the octet string $\tilde{M}_{\mathrm{rec}}$ by the following or an equivalent sequence of steps:

a) Compute the hash-token $h = \mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}})$;

b) Compute the data input $d = h \| (\mathrm{Hash}_2(h) \oplus \tilde{M}_{\mathrm{rec}})$.

NOTE 1 ECAO mandates the usage of added redundancy with the hash-token h; ECAO explicitly specifies the method for producing the data input.

NOTE 2 The above padding criteria introduce natural redundancy of more than 7 bits and close to (or equal to) 8 bits. Hence the total redundancy is about $L_{\mathrm{red}} + 1$ octets, or more than $L(n) / 2$ when $L_{\mathrm{red}} = \lfloor L(n) / 2 \rfloor$.

NOTE 3 This method is, in principle, amenable to “single-pass” processing since the non-recoverable message part M_{clr} is not processed at all.

11.3.4 Computing the signature (computations modulo *n*)

The signature $(r, s) \in \{\mathtt{0, 1}\}^{8(L_F + 1)} \times [1, n - 1]$ shall be computed by the following or an equivalent sequence of steps:

a) Compute the first part r of the signature as $r = d \oplus \Pi$;

b) Compute $u = \mathrm{MGF}(r \| M_{\mathrm{clr}})$;

c) Compute $t = \mathrm{OS2IP}(u) \bmod n$;

d) If $t = 0$, then the process of signature generation must be repeated with a new random value k;

e) Compute the second part s of the signature as $s = (k - x_A t) \bmod n$;

f) If $s = 0$, then the process of signature generation must be repeated with a new random value k;

g) Erase k.

11.3.5 Formatting the signed message

The pair $(r, s) \in \{\texttt{0}, \texttt{1}\}^{8(L_F + 1)} \times [1, n - 1]$ constitutes A's signature on the message M. The signature (r, s) and the non-recoverable message part M_{clr} constitute the signed message.

11.4 Signature verification process

11.4.1 Input and output

The input to the signature verification process consists of

— the domain parameters,

— A's public verification key Y_A, and

— the signed message.

The verifier B extracts from the signed message

— the received signature, represented as an octet string r' and an integer s', and

— the non-recoverable message part M'_{clr} (which may be null).

The output of the signature verification process is either the recovered message M' or "reject."

11.4.2 Signature size verification

Verify that $L(r') = L_F + 1$ and $0 < s' < n$; if not, then reject the signature.

11.4.3 Recovering the pre-signature (elliptic curve computations)

The pre-signature shall be recovered from the received signature (r', s') and the received non-recoverable message part M'_{clr} by the following or an equivalent sequence of steps:

a) Compute $u' = \mathrm{MGF}(r' \parallel M'_{\mathrm{clr}})$;

b) Compute $t' = \mathrm{OS2IP}(u') \bmod n$;

c) If $t' = 0$, then reject the signature;

d) Compute the elliptic curve point $R' = s'P + t'Q$;

e) If $R' = \mathrm{O}$, then reject the signature;

f) Convert R' to an octet string $\Pi' = \mathrm{EC2OSP}_E(R', \texttt{compressed})$.

11.4.4 Recovering the data input

The data input shall be recovered from the octet strings r' and Π' by the following or an equivalent sequence of steps:

a) Compute the recovered data input $d' = r' \oplus \Pi'$;

b) Compute the recovered hash-token $h' = [d']^{8L_{\mathsf{red}}}$;

c) Compute $\tilde{M}'_{\mathsf{rec}} = [d']_{8(L_F + 1 - L_{\mathsf{red}})} \oplus \mathrm{Hash}_2(h')$.

11.4.5 Checking the signature

Check the added redundancy by the following or an equivalent sequence of steps:

a) Re-compute the hash-token $h'' = \mathrm{Hash}_1(\tilde{M}'_{\mathsf{rec}})$;

b) Check whether $h' = h''$ holds or not; if not, then reject the signature.

Recover the message by the following or an equivalent sequence of steps:

a) Let pad'_1 be the leftmost non-zero octet in $\tilde{M}'_{\mathsf{rec}}$;

b) If $\mathrm{pad}'_1 \neq \mathrm{Oct}(1)$, then reject the signature;

c) Let pad'_0 and M'_{rec} be the leftmost and rightmost octets of $\tilde{M}'_{\mathsf{rec}}$, respectively, so that $\tilde{M}'_{\mathsf{rec}} = \mathrm{pad}'_0 \,\|\, \mathrm{pad}'_1 \,\|\, M'_{\mathsf{rec}}$ and $\mathrm{OS2IP}(\mathrm{pad}'_0) = 0$;

d) Compute $M' = M'_{\mathsf{rec}} \,\|\, M'_{\mathsf{clr}}$;

e) Output M'.

12 ECPV (Elliptic Curve Pintsov-Vanstone message recovery signature)[4)]

12.1 Domain and user parameters

The domain parameter specifies an additive group of order n in an elliptic curve E over an explicitly given finite field F.

The length L_{red} in octets of the added redundancy corresponds to the security parameter and is set between 1 and 255 inclusive, along with other redundancy criteria; see Clause 12.2.3.

A also uses a hash function, a key derivation function and a symmetric cipher

— $\mathrm{Hash} : \{0, 1\}^{8*} \rightarrow \{0, 1\}^{8(L(n) - 1)}$,

— $\mathrm{KDF} : \{0, 1\}^{8*} \rightarrow \{0, 1\}^{8L_{\mathsf{key}}}$, and

— $\mathrm{Sym} : \{0, 1\}^{8*} \times \{0, 1\}^{8L_{\mathsf{key}}} \rightarrow \{0, 1\}^{8*}$.

Here L_{key} denotes the length in octets of the key used with Sym. KDF is defined by $\mathrm{KDF}(x) = \mathrm{MGF2}(x, L_{\mathsf{key}})$ for $x \in \{0, 1\}^{8*}$.

4) This signature mechanism is based on a scheme defined in [10].

The keys of the ECPV signature scheme are produced as follows:

a) A's private signature key x_A which is a random integer in the interval $[1, n-1]$;

b) A's public verification key Y_A computed as in Clause 7.3.

NOTE 1 For the definition of an explicitly given finite field, see Clause A.3.

NOTE 2 For the definition of an elliptic curve over an explicitly given finite field, see Clause A.4.

NOTE 3 L_{key} corresponds to the security parameter and shall be chosen to achieve security objectives. The symmetric cipher may use exclusive-or ($\oplus$) encryption; in such case L_{key} must be equal to the length of the data input, and the maximum length of the recoverable message part shall be determined by the domain parameter; see Clause 12.2.3.

12.2 Signature generation process

12.2.1 Input and output

The input to the signature generation process consists of

— the domain parameters,

— the private signature key x_A, and

— a message M to be signed.

The output of the signature generation process is a pair $(r, s) \in \{0, 1\}^{8*} \times [1, n-1]$ that constitutes A's digital signature to the message.

12.2.2 Producing a randomizer and the pre-signature (Elliptic curve computations)

The pre-signature (the symmetric key) $\Pi \in \{0, 1\}^{8L_{\text{key}}}$ shall be computed by the following or an equivalent sequence of steps:

a) Select a random integer k in the interval $[1, n-1]$;

b) Compute the elliptic curve point $R = kP = (x, y)$;

c) Convert x to an octet string $S = \mathrm{FE2OSP}_F(x)$;

d) Compute the symmetric key $\Pi = \mathrm{KDF}(S)$.

12.2.3 Splitting the message and producing the data input

A splits the message M to the recoverable part M_{rec} as being the leftmost octets of M as agreed upon and the remaining portion of the message M_{clr}. Note that the choice of Sym may introduce a length limitation for the input. M_{rec} and M_{clr} shall be encoded and formatted properly as agreed upon by both parties. Also, a random nonce may be used in place of M_{clr}.

Form an octet string d by taking M_{rec} and the added redundancy as follows:

a) Convert L_{red} to a single octet $C_{\text{red}} = \mathrm{Oct}(L_{\text{red}})$;

b) Let $\tilde{C}_{\text{red}}$ be the octet string formed from the octet C_{red} repeated L_{red} times (thus $\tilde{C}_{\text{red}}$ shall have length L_{red});

c) Compute $d = \tilde{C}_{\text{red}} \,\|\, M_{\text{rec}}$.

NOTE 1 ECPV explicitly specifies the method for producing the data input.

NOTE 2 This method is, in principle, amenable to "single-pass" processing since the non-recoverable message part is not processed at all.

NOTE 3 In ECPV, the encoding method of the recoverable message part and the padding criteria for Sym might introduce natural redundancy for the data input and thus increase the amount of total redundancy. Normally L_{red} shall be chosen so that the total redundancy is more than $L(n) / 2$ or $L_{\text{Hash}} / 2$.

NOTE 4 ECPV can handle a recoverable message part of essentially any length in octets.

NOTE 5 In order to achieve security objectives, at least one of the following specifications is recommended for use in ECPV:

— The redundancy criteria might specify that the recoverable message part has a fixed length, or that it begins with a fixed-length representation of its length;

— The redundancy criteria might specify the use of a DER encoding of an ASN.1 type for the recoverable message part;

— The domain parameter might specify that the non-recoverable message part has a fixed length (perhaps empty), or that it ends with a fixed-length representation of its length.

12.2.4 Computing the signature (Computations modulo n)

The signature $(r, s) \in \{0, 1\}^{8*} \times [1, n-1]$ shall be computed by the following or an equivalent sequence of steps:

a) Compute $r = \text{Sym}(d, \Pi)$;

b) Compute $u = \text{Hash}(r \,||\, M_{\text{clr}})$;

c) Convert $t = \text{OS2IP}(u)$; note that $t \in [0, n-1]$;

d) If $t = 0$, then the process of signature generation must be repeated with a new random value k;

e) Compute $s = (k - x_A t) \bmod n$;

f) If $s = 0$, then the process of signature generation must be repeated with a new random value k;

g) Erase k.

Output the signature (r, s) and the partial message part M_{clr} (which may be null).

12.2.5 Formatting the signed message

The pair $(r, s) \in \{0, 1\}^{8*} \times [1, n-1]$ constitutes A's signature on the message M.

12.3 Signature verification process

12.3.1 Input and output

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of

— the domain parameters,

— A's public verification key Y_A,

— the received signature for M, represented as an octet string r' and an integer s', and

— the non-recoverable message M'_{clr} (if any).

To verify A's signature for M, B executes the steps described in Clauses 12.3.2 through 12.3.5.

12.3.2 Signature size verification

Verify that $0 < s' < n$; if not, then reject the signature.

12.3.3 Recovering the pre-signature (Elliptic curve computations)

The pre-signature (the symmetric key) shall be recovered from the signature by the following or an equivalent sequence of steps:

a) Compute $u' = \mathrm{Hash}(r' \parallel M'_{\mathrm{clr}})$;

b) Convert $t' = \mathrm{OS2IP}(u')$; note that $t' \in [0, n-1]$;

c) If $t' = 0$, then reject the signature;

d) Compute $R' = s'P + t'Q = (x', y')$ and perform the following operations:

   1) If R' is the point at infinity, then reject the signature;

   2) Otherwise compute the symmetric key $\Pi' = \mathrm{KDF}(\mathrm{FE2OSP}_F(x'))$.

12.3.4 Recovering the data input or the message

The data input shall be recovered by computing $d' = \mathrm{Sym}^{-1}(r', \Pi')$, where Sym^{-1} denotes the decryption function of the symmetric cipher Sym.

12.3.5 Checking the signature

Verify the added redundancy of d' and recover M'_{rec} by the following or an equivalent sequence of steps:

a) If $L(d') < L_{\mathrm{red}}$, then reject the signature;

b) Convert L_{red} to a single octet $C_{\mathrm{red}} = \mathrm{Oct}(L_{\mathrm{red}})$;

c) Let $\tilde{C}_{\mathrm{red}}$ be the octet string formed from the octet C_{red} repeated L_{red} times (thus $\tilde{C}_{\mathrm{red}}$ shall have length L_{red});

d) Check the added redundancy by $\tilde{C}_{\mathrm{red}} = [d']^{8L_{\mathrm{red}}}$; if it does not hold, then reject the signature;

e) Compute $M'_{\mathrm{rec}} = [d']_{8(L(d') - L_{\mathrm{red}})}$;

f) Check the natural redundancy of M'_{rec} in accordance with its encoding and formatting methods; if it is not satisfied, then reject the signature;

g) Check the format of M'_{clr} (if any); if it is not satisfied, then reject the signature;

h) Recover M' as the following:

   1) In case M'_{clr} is either the null string or a random nonce, set $M' = M'_{\mathrm{rec}}$;

   2) Otherwise, recover M' from M'_{rec} and M'_{clr};

i) Output M'.

13 ECKNR (Elliptic Curve KCDSA/Nyberg-Rueppel message recovery signature)[5)]

13.1 Domain parameter and user keys

The domain parameter specifies an additive group of order n in an elliptic curve E over an explicitly given finite field F.

A also uses a mask generation function:

— $\mathrm{MGF} : \{\mathtt{0},\mathtt{1}\}^{8*} \rightarrow \{\mathtt{0},\mathtt{1}\}^{8L(n)}$.

The function MGF is defined as $\mathrm{MGF}(x) = \mathrm{MGF2}(x, L(n))$ for $x \in \{\mathtt{0},\mathtt{1}\}^{8*}$ with the underlying hash-function Hash. The keys of the ECKNR signature scheme are produced as follows:

a) A's private signature key x_A which is a random integer in the interval $[1, n-1]$;

b) A's public verification key Y_A computed as in Clause 7.3.

A's certification-derived data z_A is defined as $z_A = [\mathrm{Cert}_A]^{L\mathrm{B,\,Hash}}$, where Cert_A denotes the certification data of A, that is A's public verification key Y_A converted to a bit string. When $Y_A = (x_0, y_0)$, $\mathrm{Cert}_A = \mathrm{FE2OSP}_F(x_0) \,\|\, \mathrm{FE2OSP}_F(y_0)$ and $L_{\mathrm{B,\,Hash}}$ is the bit length of input size of hash function. For example, $L_{\mathrm{B,\,Hash}}$ in RIPEMD-160 becomes 512.

NOTE 1 For the definition of an explicitly given finite field, see Clause A.3.

NOTE 2 For the definition of an elliptic curve over an explicitly given finite field, see Clause A.4.

13.2 Signature generation process

13.2.1 Input and output

The input to the signature process consists of:

— the domain parameters;

— A's private signature key x_A;

— A's certification data z_A, and

— the message M to be signed, which is split to the recoverable part M_{rec} as being the leftmost octets of M as agreed upon and the remaining portion of the message M_{clr}.

The output of the signature generation process is a pair $(r, s) \in \{\mathtt{0},\mathtt{1}\}^{8L(n)} \times [1, n-1]$ that constitutes A's digital signature to the message M.

13.2.2 Producing a randomizer and the pre-signature (elliptic curve computations)

The pre-signature $\Pi \in \{\mathtt{0},\mathtt{1}\}^{8L(n)}$ shall be computed by the following or an equivalent sequence of steps:

a) Select a random integer k in the interval $[1, n-1]$;

b) Compute the elliptic curve point $R = kP$;

c) Convert R into an octet string and compute the hash value $\Pi = \mathrm{MGF}\,(\mathrm{EC2OSP}_E(R, \mathtt{compressed}))$.

NOTE For the definition of the conversion function EC2OSP with the format specifier `compressed`, see Clause B.6.

5) This signature mechanism is based on a scheme defined in [6] and [11].

13.2.3 Producing the data Input

The data d with added or natural redundancy in $\{0, 1\}^{8L(n)}$ is produced from a message, see Clauses 7.4.3.

13.2.4 Computing the signature (computations modulo n)

The signature $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n - 1]$ shall be computed by the following or an equivalent sequence of steps:

a) Compute the first part of A's signature $r = d \oplus \Pi \oplus \mathrm{MGF}(z_A \parallel M_{\mathrm{clr}})$;

b) Set $t = \mathrm{OS2IP}(r) \bmod n$;

c) Compute the second part of A's signature $s = (k - x_A t) \bmod n$;

d) Erase k.

If the signature generation process yields r such that $\mathrm{OS2IP}(r) = 0 \bmod n$ or $s = 0$, then the process of signature generation must be repeated with a new random value k.

13.2.5 Formatting the signed message

The pair $(r, s) \in \{0, 1\}^{8L(n)} \times [1, n - 1]$ constitutes A's signature on the message M.

13.3 Signature verification process

13.3.1 Input and output

The signature verification process consists of three steps: calculation of the message digest, elliptic curve computations, and signature checking.

The input to the signature verification process consists of:

— the domain parameters;

— A's public verification key Y_A;

— A's certification data z_A;

— the received signature for M, represented as an octet string r' and an integer s', and

— the non-recoverable message part M'_{clr} (if any).

The output of the signature verification process is either the recovered data input d' or "reject."

13.3.2 Signature size verification

Verify that $r' \in \{0, 1\}^{8L(n)}$, $\mathrm{OS2IP}(r') \neq 0$ and $0 < s' < n$; if any one of these conditions is not satisfied, then reject the signature.

13.3.3 Recovering the pre-signature (elliptic curve computations)

The pre-signature shall be recovered from the received signature (r', s') by the following or an equivalent sequence of steps:

a) Set $t' = \mathrm{OS2IP}(r') \bmod n$;

b) Compute the elliptic curve point $R' = s'P + t'Q$;

c) Convert R' into an octet string and compute the hash value $\Pi' = \mathrm{MGF}(\mathrm{EC2OSP}_E(R', \texttt{compressed}))$.

13.3.4 Recovering the data input or the message

The data input shall be recovered from the first part of the received signature r' and the recovered pre-signature Π' by the following or an equivalent sequence of steps:

a) Compute the recovered data input $d' = r' \oplus \Pi' \oplus \mathrm{MGF}(z_A \,||\, M'_{\mathrm{clr}})$.

13.3.5 Checking the signature

Check the redundancy. If it is correct, output d', otherwise reject.

Annex A
(informative)

Mathematical conventions

A.1 Bit strings

A bit is either zero "0" or one "1." A bit string x is a finite sequence $\langle x_{l-1}, \ldots, x_0 \rangle$ of bits $x_0, \ldots, x_{l-1}$. The ***length of a bit string*** x is the number l of bits in the string x. Given a non-negative integer n, $\{0, 1\}^n$ denotes the set of bit strings of length n. $\{0, 1\}^* = \cup_{n \geq 0} \{0, 1\}^n$ denotes the set of bit strings, including the null string (whose length is 0).

A.2 Octet strings

An octet is a bit string of length 8. An octet string is a finite sequence of octets. The ***length of an octet string*** is the number of octets in the string. $\{0, 1\}^{8*}$ denotes the set of octet strings, including the null string (whose length is 0). An octet is often written in its hexadecimal format, using the range between `00` and `FF`; see Clause B.3.

A.3 Finite fields

This clause describes a very general framework for describing specific finite fields. A finite field specified in this way is called an ***explicitly given finite field***, and it is determined by ***explicit data***. For a finite field F of cardinality $q = p^e$, where p is prime and $e \geq 1$, explicit data for F consists of p and e, along with a "multiplication table," which is a matrix $T = (T_{ij})_{1 \leq i,j \leq e}$, where each T_{ij} is an e-tuple over $[0, p - 1]$.

The set of elements of F is the set of all e-tuples over $[0, p - 1]$. The entries of T are themselves viewed as elements of F.

Addition in F is defined element-wise: if

$$a = (a_1, \ldots, a_e) \in F \text{ and } b = (b_1, \ldots, b_e) \in F,$$

then $a + b = c$, where

$$c = (c_1, \ldots, c_e) \text{ and } c_i = (a_i + b_i) \bmod p \ (1 \leq i \leq e).$$

A scalar multiplication operation for F is also defined element-wise: if

$$a = (a_1, \ldots, a_e) \in F \text{ and } d \in [0, p - 1],$$

then $d \cdot a = c$, where

$$c = (c_1, \ldots, c_e) \text{ and } c_i = (d \cdot a_i) \bmod p \ (1 \leq i \leq e).$$

Multiplication in F is defined via the multiplication table T, as follows: if

$$a = (a_1, \ldots, a_e) \in F \text{ and } b = (b_1, \ldots, b_e) \in F,$$

$$a \cdot b = \sum_{1 \leq i \leq e} \sum_{1 \leq j \leq e} (a_i b_j \bmod p) T_{ij},$$

where the products $(a_i b_j \bmod p)T_{ij}$ are defined using the above rule for scalar multiplication, and where these products are summed using the above rule for addition in F. It is assumed that the multiplication table defines an algebraic structure that satisfies the usual axioms of a field; in particular, there exist additive and multiplicative identities, every element has an additive inverse, and every element besides the additive identity has a multiplicative inverse.

Observe that the additive identity of F, denoted 0_F, is the all-zero e-tuple, and that the multiplicative identity of F, denoted 1_F, is a non-zero e-tuple whose precise format depends on T.

NOTE 1 The field F is a vector space of dimension e over the prime field F' of cardinality p, where scalar multiplication is defined as above. The prime p is called the characteristic of F. For $1 \le i \le e$, let θ_i denote the e-tuple over F' whose i-th component is 1, and all of whose other components are 0. The elements $\theta_1, \ldots, \theta_e$ form an ordered basis of F as a vector space over F'. Note that for $1 \le i, j \le e$, we have $\theta_i \cdot \theta_j = T_{ij}$.

NOTE 2 For $e > 1$, two types of standard bases are defined that are commonly used in implementations of finite field arithmetic:

— $\theta_1, \ldots, \theta_e$ is called a ***polynomial basis*** for F over F' if for some $\theta \in F$, $\theta_i = \theta^{e-i}$ for $1 \le i \le e$. Note that in this case, $1_F = \theta_e$; and

— $\theta_1, \ldots, \theta_e$ is called a ***normal basis*** for F over F' if for some $\theta \in F$, $\theta_i = \theta^{p^{i-1}}$ for $1 \le i \le e$. Note that in this case, $1_F = c\sum_{1 \le i \le e} \theta_i$ for some $c \in [0, p-1]$; if $p = 2$, then the only possible choice for c is 1; moreover, one can always choose a normal basis for which $c = 1$.

A.4 Elliptic curves

An elliptic curve E over an explicitly given finite field F is a set of points $P = (x, y)$, where x and y are elements of F that satisfy a certain equation, together with the “point at infinity,” denoted by $\mathcal{O}$. For the purposes of this part of ISO/IEC 9796, the curve E is specified by two field elements $a, b \in F$, called the ***coefficients*** of E.

Let p be the characteristic of F.

If $p > 3$, then a and b shall satisfy $4a^3 + 27b^2 \ne 0_F$, and every point $P = (x, y)$ on E (other than $\mathcal{O}$) shall satisfy the equation

$$y^2 = x^3 + ax + b.$$

If $p = 2$, then b shall satisfy $b \ne 0_F$, and every point $P = (x, y)$ on E (other than $\mathcal{O}$) shall satisfy the equation

$$y^2 + xy = x^3 + ax^2 + b.$$

If $p = 3$, then a and b shall satisfy $a \ne 0_F$ and $b \ne 0_F$, and every point $P = (x, y)$ on E (other than $\mathcal{O}$) shall satisfy the equation

$$y^2 = x^3 + ax^2 + b.$$

The points on an elliptic curve form a finite abelian group, where $\mathcal{O}$ is the identity element. There exist efficient algorithms to perform the group operation of an elliptic curve, but the implementation of such algorithms is out of the scope of this part of ISO/IEC 9796.

NOTE See ISO/IEC 15946-1 for more information on how to efficiently implement elliptic curve group operations.

Annex B
(normative)

Conversion functions

B.1 Octet string / bit string conversion: OS2BSP and BS2OSP

Primitives OS2BSP and BS2OSP that convert between octet strings and bit strings are defined as follows:

— The function $\mathrm{OS2BSP}(x)$ takes as input an octet string x and outputs x, which is also a bit string; and

— The function $\mathrm{BS2OSP}(y)$ takes as input a bit string y, whose length is a multiple of 8, and outputs the unique octet string x such that $y = \mathrm{OS2BSP}(x)$.

B.2 Bit string / integer conversion: BS2IP and I2BSP

Primitives BS2IP and I2BSP that convert between bit strings and integers are defined as follows:

— The function $\mathrm{BS2IP}(x)$ maps a bit string x to an integer value x', as follows. If $x = \langle x_{l-1}, \ldots, x_0 \rangle$ where $x_0, \ldots, x_{l-1}$ are bits, then the value x' is defined as $x' = \sum_{0 \le i < l,\, x_i = \text{‘}1\text{’}} 2^i$; and

— The function $\mathrm{I2BSP}(m, l)$ takes as input two non-negative integers m and l, and outputs the unique bit string x of length l such that $\mathrm{BS2IP}(x) = m$, if such an x exists. Otherwise, the function fails.

The ***length in bits of a non-negative integer*** n is the number of bits in its binary representation, i.e., $\lceil \log_2(n + 1) \rceil$. As a notational convenience, $\mathrm{Oct}(m)$ is defined as $\mathrm{Oct}(m) = \mathrm{I2BSP}(m, 8)$.

NOTE Note that $\mathrm{I2BSP}(m, l)$ fails if and only if the length of m in bits is greater than l.

B.3 Octet string / integer conversion: OS2IP and I2OSP

Primitives OS2IP and I2OSP that convert between octet strings and integers are defined as follows:

— The function $\mathrm{OS2IP}(x)$ takes as input an octet string, and outputs the integer $\mathrm{BS2IP}(\mathrm{OS2BSP}(x))$; and

— The function $\mathrm{I2OSP}(m, l)$ takes as input two non-negative integers m and l, and outputs the unique octet string x of length l such that $\mathrm{OS2IP}(x) = m$, if such an x exists. Otherwise, the function fails.

The ***length in octets of a non-negative integer*** n is the number of digits in its representation base 256, i.e., $\lceil \log_{256}(n + 1) \rceil$; this quantity is denoted $L(n)$.

NOTE 1 Note that $\mathrm{I2OSP}(m, l)$ fails if and only if the length of m in octets is greater than l.

NOTE 2 An octet x is often written as $\mathrm{OS2IP}(x)$ in its hexadecimal format of length 2; when $\mathrm{OS2IP}(x) < 16$, “0”, representing the bit string 0000, is prepended.

B.4 Finite field element / integer conversion: FE2IP$_F$

The primitive $\mathrm{FE2IP}_F$ that converts elements of F to integer values is defined as follows:

— The function $\mathrm{FE2IP}_F$ maps an element $a \in F$ to an integer value a', as follows. If the cardinality of F is $q = p^e$, where p is prime and $e \ge 1$, then an element a of F is an e-tuple $(a_1, \ldots, a_e)$, where $a_i \in [0 \,..\, p)$ for $1 \le i \le e$, and the value a' is defined as $a' = \sum_{1 \le i \le e} a_i p^{i-1}$;

B.5 Octet string / finite field element conversion: $\mathrm{OS2FEP}_F$ and $\mathrm{FE2OSP}_F$

Primitives $\mathrm{OS2FEP}_F$ and $\mathrm{FE2OSP}_F$ that convert between octet strings and elements of an explicitly given finite field F are defined as follows:

— The function $\mathrm{FE2OSP}_F(a)$ takes as input an element a of the field F and outputs the octet string $\mathrm{I2OSP}(a', l)$, where $a' = \mathrm{FE2IP}_F(a)$, and l is the length in octets of $|F|-1$, i.e., $l = \lceil \log_{256}|F| \rceil$. Thus, the output of $\mathrm{FE2OSP}_F(a)$ is always an octet string of length exactly $\lceil \log_{256}|F| \rceil$; and

— The function $\mathrm{OS2FEP}_F(x)$ takes as input an octet string x, and outputs the (unique) field element $a \in F$ such that $\mathrm{FE2OSP}_F(a) = x$, if such an a exists, and otherwise fails.

Note that $\mathrm{OS2FEP}_F(x)$ fails if and only if either x does not have length exactly $\lceil \log_{256}|F| \rceil$, or $\mathrm{OS2IP}(x) \geq |F|$; this quantity is denoted L_F.

B.6 Elliptic curve / octet string conversion: $\mathrm{EC2OSP}_E$ and $\mathrm{OS2ECP}_E$

B.6.1 Compressed elliptic curve points

Let E be an elliptic curve over an explicitly given finite field F, where F has characteristic p.

A point $P \neq O$ can be represented in either `compressed`, `uncompressed`, or `hybrid` form.

If $P = (x, y)$, then (x, y) is the ***uncompressed form*** of P.

Let $P = (x, y)$ be a point on the curve E, as above. The ***compressed form*** of P is the pair $(x, \tilde{y})$, where $\tilde{y} \in \{0, 1\}$ is determined as follows:

— If $p \neq 2$ and $y = 0_F$, then $\tilde{y} = 0$;

— If $p \neq 2$ and $y \neq 0_F$, then $\tilde{y} = ((y'/p^f) \bmod p) \bmod 2$, where $y' = \mathrm{FE2IP}_F(y)$, and where f is the largest non-negative integer such that $p^f \mid y'$;

— If $p = 2$ and $x = 0_F$, then $\tilde{y} = 0$; and

— If $p = 2$ and $x \neq 0_F$, then $\tilde{y} = \lfloor z'/2^f \rfloor \bmod 2$, where $z = y/x$, where $z' = \mathrm{FE2IP}_F(z)$, and where f is the largest non-negative integer such that 2^f divides $\mathrm{FE2IP}_F(1_F)$.

The ***hybrid form*** of $P = (x, y)$ is the triple $(x, \tilde{y}, y)$, where $\tilde{y}$ is as in the previous paragraph.

B.6.2 Point decompression algorithms

There exist efficient procedures for ***point decompression***, i.e., computing y from $(x, \tilde{y})$. These are briefly described here:

— Assume $p \neq 2$, and let $(x, \tilde{y})$ be the compressed form of (x, y). The point (x, y) satisfies an equation $y^2 = f(x)$ for a polynomial $f(x)$ over F in x. If $f(x) = 0_F$, then there is only one possible choice for y, namely, $y = 0_F$. Otherwise, if $f(x) \neq 0$, then there are two possible choices of y, which differ only in sign, and the correct choice is determined by $\tilde{y}$. There are well-known algorithms for computing square roots in finite fields, and so the two choices of y are easily computed; and

— Assume $p = 2$, and let $(x, \tilde{y})$ be the compressed form of (x, y). The point (x, y) satisfies an equation $y^2 + xy = x^3 + ax^2 + b$. If $x = 0_F$, then we have $y^2 = b$, from which y is uniquely determined and easily computed. Otherwise, if $x \neq 0_F$, then setting $z = y/x$, we have $z^2 + z = g(x)$, where $g(x) = (x + a + bx^{-2})$. The value of y is uniquely determined by, and easily computed from, the values z and x, and so it suffices to compute z. To compute z, observe that for a fixed x, if z is one solution to the equation $z^2 + z = g(x)$, then there is exactly one other solution, namely $z + 1_F$. It is easy to compute these two candidate values of z, and the correct choice of z is easily seen to be determined by $\tilde{y}$.

B.6.3 Conversion functions

Let E be an elliptic curve over an explicitly given finite field F.

Primitives $\mathrm{EC2OSP}_E$ and $\mathrm{OS2ECP}_E$ for converting between points on an elliptic curve E and octet strings are defined as follows:

a) The function $\mathrm{EC2OSP}_E(P, \mathrm{fmt})$ takes as input a point P on E and a format specifier fmt, which is one of the symbolic values `compressed`, `uncompressed`, or `hybrid`. The output is an octet string EP, computed as follows:

   1) If $P = \mathcal{O}$, then $\mathrm{EP} = \mathrm{Oct}(0)$; and

   2) If $P = (x, y) \neq \mathcal{O}$, with compressed form $(x, \tilde{y})$, then $\mathrm{EP} = H \parallel X \parallel Y$, where

      i) H is a single octet of the form $\mathrm{Oct}(4U + C \cdot (2 + \tilde{y}))$, where

         I) $U = 1$ if fmt is either `uncompressed` or `hybrid`, and otherwise, $U = 0$, and

         II) $C = 1$ if fmt is either `compressed` or `hybrid`, and otherwise, $C = 0$,

      ii) X is the octet string $\mathrm{FE2OSP}_F(x)$, and

      iii) Y is the octet string $\mathrm{FE2OSP}_F(y)$ if fmt is either `uncompressed` or `hybrid`, and otherwise Y is the null octet string; and

b) The function $\mathrm{OS2ECP}_E(\mathrm{EP})$ takes as input an octet string EP. If there exists a point P on the curve E and a format specifier fmt such that $\mathrm{EC2OSP}_E(P, \mathrm{fmt}) = \mathrm{EP}$, then the function outputs P (in uncompressed form), and otherwise, the function fails. Note that the point P, if it exists, is uniquely defined, and so the function $\mathrm{OS2ECP}_E(\mathrm{EP})$ is well defined.

NOTE If the format specifier fmt is `uncompressed`, then the value $\tilde{y}$ need not be computed.

Annex C
(normative)

Mask generation functions (Key derivation functions)

This annex describes "mask generation functions" that are referred to in this part of ISO/IEC 9796. Specific implementations of mask generation functions that are allowed for use in this part of ISO/IEC 9796 are specified.

A mask generation function is a function $\mathrm{MGF}^*(x, l)$ that takes as input an octet string x and an integer l, and outputs an octet string of length l. The string x is of arbitrary length, although an implementation may define a (very large) maximum length for x and maximum size for l.

NOTE In some other documents and standards, the term "key derivation function" is used instead of "mask generation function."

C.1 Allowable mask generation functions

The mask generation functions that are allowed in this part of ISO/IEC 9796 are $\mathrm{MGF1}$, described below in Clause C.2, and $\mathrm{MGF2}$, described below in Clause C.3.

C.2 MGF1

$\mathrm{MGF1}$ is a family of mask generation functions, parameterized by the following system parameter:

— Hash: a hash-function.

For an octet string x and a non-negative integer l, $\mathrm{MGF1}(x, l)$ is defined to be

$$[\mathrm{Hash}(x \,\|\, \mathrm{I2OSP}(0, 4)) \,\|\, \mathrm{Hash}(x \,\|\, \mathrm{I2OSP}(1, 4)) \,\|\, \cdots \,\|\, \mathrm{Hash}(x \,\|\, \mathrm{I2OSP}(k-1, 4))]^{8l},$$

where $k = \lceil l / L_{\mathrm{Hash}} \rceil$.

C.3 MGF2

$\mathrm{MGF2}$ is a family of mask generation functions, parameterized by the following system parameter:

— Hash: a hash-function.

For an octet string x and a non-negative integer l, $\mathrm{MGF2}(x, l)$ is defined to be

$$[\mathrm{Hash}(x \,\|\, \mathrm{I2OSP}(1, 4)) \,\|\, \mathrm{Hash}(x \,\|\, \mathrm{I2OSP}(2, 4)) \,\|\, \cdots \,\|\, \mathrm{Hash}(x \,\|\, \mathrm{I2OSP}(k, 4))]^{8l},$$

where $k = \lceil l / L_{\mathrm{Hash}} \rceil$.

NOTE $\mathrm{MGF2}$ is the same as $\mathrm{MGF1}$, except that the counter runs from 1 to k, rather than from 0 to $k-1$.

Annex D
(informative)

Example method for producing the data input

In this annex, an example method for producing the data input with added redundancy and for checking the redundancy in Clauses 7.4.3 and 7.5.4 is described. This method can be combined with the following schemes described in this part of ISO/IEC 9796: NR, ECNR, ECMR and ECKNR.

D.1 Splitting the message and producing the data input

A selects a hash-function $\mathrm{Hash} : \{0, 1\}^{8*} \to \{0, 1\}^{8L_{\mathrm{red}}}$. A also specifies the use of hash-function identifier option; A sets $L_{\mathrm{HashID}} = 1$ when hash-function identification is desired and $L_{\mathrm{HashID}} = 0$ otherwise. A sets an octet string $\mathrm{trailer}$ to be the hash-function identifier when $L_{\mathrm{HashID}} = 1$ and to be the null octet otherwise. These information must be provided as domain parameters. Also, each mechanism specifies the length of the data input; let L_{dat} be the length of the data input in octets.

The data input d is then produced from a message M by the following or an equivalent sequence of steps:

a) Compute the maximum length L_{max} of recoverable part as $L_{\mathrm{max}} = L_{\mathrm{dat}} - L_{\mathrm{red}} - L_{\mathrm{HashID}}$;

b) Split the message M to the recoverable part M_{rec} as being the leftmost octets of M and the remaining portion of the message M_{clr}, as follows:

   1) If $L(M) \le L_{\mathrm{max}}$, then set $M_{\mathrm{rec}} = M$ and $M_{\mathrm{clr}} = \varnothing$ (the null string);

   2) If $L(M) > L_{\mathrm{max}}$, then split M into M_{rec} and M_{clr} such as $M = M_{\mathrm{rec}} \,||\, M_{\mathrm{clr}}$ satisfying $L_{\mathrm{max}} > L(M_{\mathrm{rec}})$;

c) Convert the lengths to octet strings $C_{\mathrm{rec}} = \mathrm{I2OSP}(L_{\mathrm{rec}}, 8)$ and $C_{\mathrm{clr}} = \mathrm{I2OSP}(L_{\mathrm{clr}}, 8)$;

d) Compute the hash-token $h \in \{0, 1\}^{8L_{\mathrm{red}} + 8L_{\mathrm{HashID}}}$ as $h = \mathrm{Hash}(C_{\mathrm{rec}} \,||\, C_{\mathrm{clr}} \,||\, M_{\mathrm{rec}} \,||\, M_{\mathrm{clr}} \,||\, \Pi) \,||\, \mathrm{trailer}$;

e) Compute the padding string $\mathrm{pad} = \mathrm{I2OSP}(0, L_{\mathrm{max}} - L_{\mathrm{rec}})$;

f) Produce the data input $d \in \{0, 1\}^{8L_{\mathrm{dat}}}$ as $d = \mathrm{pad} \,||\, h \,||\, M_{\mathrm{rec}}$.

A must include the length L_{rec} in the signed message, along with the signature (r, s) and the non-recoverable message part M_{clr}.

D.2 Checking the redundancy

B receives a signed message consisting of the first part r' of the signature, the second part s' of the signature, the recoverable part length L'_{rec} and the non-recoverable message part M'_{clr}. The pre-signature $\Pi' \in \{0, 1\}^{8*}$ and the data input $d' \in \{0, 1\}^{8L_{\mathrm{dat}}}$ is recovered from the received signature (r', s').

B verifies the signature and recovers the message by the following or an equivalent sequence of steps:

a) Compute $L_{\mathrm{max}} = L_{\mathrm{dat}} - L_{\mathrm{red}} - L_{\mathrm{HashID}}$;

b) Check if $L'_{\mathrm{rec}} \in [0, L_{\mathrm{max}}]$; if not, then reject the signature;

c) Recover the padding string, the hash-token and the recoverable part as $\mathrm{pad}' = [d']^{L_{\mathrm{max}} - L'_{\mathrm{rec}}}$, $h' = [[d']_{8L_{\mathrm{red}} + 8L_{\mathrm{HashID}} + 8L'_{\mathrm{rec}}}]^{8L_{\mathrm{red}} + 8L_{\mathrm{HashID}}}$ and $M'_{\mathrm{rec}} = [d']_{8L'_{\mathrm{rec}}}$, respectively;

d) Check the padding: if $\mathrm{OS2IP}(\mathrm{pad}') = 0$ does not hold, then reject the signature;

e) Compute the length $L'_{\mathrm{clr}} = L(M'_{\mathrm{clr}})$;

f) Convert the lengths to octet strings $C'_{\mathrm{rec}} = \mathrm{I2OSP}(L'_{\mathrm{rec}}, 8)$ and $C'_{\mathrm{clr}} = \mathrm{I2OSP}(L'_{\mathrm{clr}}, 8)$;

g) Re-compute the hash-token $h'' = \mathrm{Hash}(C'_{\mathrm{rec}} \parallel C'_{\mathrm{clr}} \parallel M'_{\mathrm{rec}} \parallel M'_{\mathrm{clr}} \parallel \Pi') \parallel \mathrm{trailer}$;

h) Check the added redundancy: if $h' = h''$ does not hold, then reject the signature;

i) Output $M'_{\mathrm{rec}} \parallel M'_{\mathrm{clr}}$.

Annex E
(normative)

ASN.1 module

E.1 Formal definition

This annex defines an ASN.1 module containing abstract syntax for the digital signature with message recovery mechanisms specified in this part of ISO/IEC 9796.

```
MessageRecoverySignatureMechanisms {
   iso(1) standard(0) signature-schemes(9796) part(3) asn1-module(1)
      message-recovery-signature-mechanisms(0)
}
DEFINITIONS EXPLICIT TAGS ::= BEGIN

IMPORTS

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

OID ::= OBJECT IDENTIFIER  -- alias

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

signatureMechanism OID ::= {
   iso(1) standard(0) hash-functions(9796) part3(3) mechanism(0)
}

MessageRecovery ALGORITHM ::= {
   dswmr-nr     |
   dswmr-ecmr   |
   dswmr-ecao   |
   dswmr-ecknr  |
   dswmr-ecpv   |
   dswmr-ecnr,

   ... -- Expect additional algorithms --
}

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

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

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

dswmr-ecknr ALGORITHM ::= {
   OID ecknr PARMS HashFunctions
}
dswmr-ecpv ALGORITHM ::= {
   OID ecpv PARMS HashFunctions
}
dswmr-ecnr ALGORITHM ::= {
   OID ecnr PARMS HashFunctions
```



```
}

-- Cryptographic algorithm identification –

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

-- Message recovery signature mechanisms --

nr    OID ::= { signatureMechanism nr(0) }
ecmr  OID ::= { signatureMechanism ecmr(1) }
ecao  OID ::= { signatureMechanism ecao(2) }
ecknr OID ::= { signatureMechanism ecknr(3) }
ecpv  OID ::= { signatureMechanism ecpv(4) }
ecnr  OID ::= { signatureMechanism ecnr(5) }

END  -- MessageRecoverySignatureMechanisms –
```

E.2 Use of subsequent object identifiers

Any one of the signature schemes uses a hash-function. Therefore a subsequent object identifier may follow for referring to a hash-function (e.g., one of the dedicated hash-functions specified in ISO/IEC 10118-3).

Annex F
(informative)

Numerical examples

F.1 Numerical examples for NR

NOTE 1 Throughout Clause F.1 we refer to ASCII encoding of data strings; this is equivalent to coding using ISO 646.

NOTE 2 Clauses F.1.2, F.1.3 and F1.4 use the domain parameter, the user keys, the randomizer and the message described in Clause F.1.1.

F.1.1 Example with partial recovery

P	ffffffff ffffffff c90fdaa2 2168c234 c4c6628b 80dc1cd1 29024e08 8a67cc74 020bbea6 3b139b22 514a0879 8e3404dd ef9519b3 cd3a431b 302b0a6d f25f1437 4fe1356d 6d51c245 e485b576 625e7ec6 f44c42e9 a637ed6b 0bff5cb6 f406b7ed ee386bfb 5a899fa5 ae9f2411 7c4b1fe6 49286651 ece65381 ffffffff ffffffff
Q	7fffffff ffffffff e487ed51 10b4611a 62633145 c06e0e68 94812704 4533e63a 0105df53 1d89cd91 28a5043c c71a026e f7ca8cd9 e69d218d 98158536 f92f8a1b a7f09ab6 b6a8e122 f242dabb 312f3f63 7a262174 d31bf6b5 85ffae5b 7a035bf6 f71c35fd ad44cfd2 d74f9208 be258ff3 24943328 f67329c0 ffffffff ffffffff
Length of Q	1023 bits
G	2
Signature key x_A	fcf63b30 a349edc2 b135b0d4 fbcf2900 8c9de512 d033dbd5 32e513c3 b2501ef7 c3bae4a5 f1368abc 4f5643e1 0f737660 c9aa959f 8362bc82 7771f89e 88a1bcbc 3276d52b 3e1ab0fc f398c937 9370241e 66b87ef9 78555971 3282a0ac 7ca11239 976f6605 29b4bc4c 7d0c9412 9ac52410 3a0eed44 f1aaa99f b1791059 0378b037
Verification key Y_A	a544638a d770ce35 c5286db8 3c124a77 f382bc7c ed585501 371928f8 1bc5e61f da841361 08beab18 e84f46d6 5cd0a9f2 4a00998d 37312a2e f28f7370 b95ce7ff 2cee0be9 1457beb0 9fe790f1 e31de199 1ca3b8db 7de3f13c 8add8e02 5eaa7a41 3ee276da 364bf447 52022ca5 48133f7c 57e94a0c 20cbff8e 98660f98 e034fe4c
Randomizer k	1698cc3 2a59174b 93511339 528fb5d8 ba386493 85630f0a 9624f5ab 71a5ccf9 29c63f3e 0e36a339 207685a4 12cec6a4 3f0ae734 bfd30703 83109786 101b036d e83b4954 048217c2 6d76a398 f7afd556 9e1cf908 091be435 de10c379 35aa8896 ee34df2a 1b29866f 29256ea5 8e2c2558 0cd65489 99579211 c5aad05f ddbda767
Pre-signature Π	0ebc1b0b baf3c121 ff29d858 7c35e42b 5614ff11 aa40ceed 454c57b6 dd3a7ee0 7732420e 7c8c7b18 2c7aaccc 52c798c0 2ec6e2bc bb67256e 032c0e13 2eaa8ca8 1dab8404 73e81f61 912827b6 23d65fac 29f5414a 2ce7ce88 07fe6891 c58aaf05 e8546e83 196b0f62 6873befe 51c0b7e3 b8ac49b2 5f416791 e0dacc23 f41f25d5
Message to be signed	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
M	41424344 45464748 494a4b4c 4d4e4f50 51525354 55565758 595a6162 63646566 6768696a 6b6c6d6e 6f707172 73747576 7778797a


```
30313233 34353637 38394142 43444546 4748494a 4b4c4d4e 4f505152
53545556 5758595a 61626364 65666768 696a6b6c 6d6e6f70 71727374
75767778 797a3031 32333435 36373839 41424344 45464748 494a4b4c
4d4e4f50 51525354 55565758 595a6162 63646566 6768696a 6b6c6d6e
6f707172 73747576 7778797a 30313233 34353637 38394142 43444546
4748494a 4b4c4d4e 4f505152 53545556 5758595a 61626364 65666768
696a6b6c 6d6e6f70 71727374 75767778 797a3031 32333435 36373839
```

F.1.2 Example with Dedicated Hash-Function 3 (otherwise known as SHA1) of ISO/IEC 10118-3

Length of hash-token	21 octets
Recoverable length L_{rec}	00000000 0000006a
Non-recoverable length L_{clr}	00000000 0000008e
Hash-code	005e4e9b e8c9a202 80ffab58 d9927041 80dcc44d
Hash-function identifier	33
Data input d	5e4e 9be8c9a2 0280ffab 58d99270 4180dcc4 4d334142 43444546 4748494a 4b4c4d4e 4f505152 53545556 5758595a 61626364 65666768 696a6b6c 6d6e6f70 71727374 75767778 797a3031 32333435 36373839 41424344 45464748 494a4b4c 4d4e4f50 51525354 55565758 595a6162 63646566 6768696a 6b6c6d6e 6f707172
First part of signature r	0ebc795a 56dc8ac4 01aad803 d50f769b 9795dbd5 f774102f 88909cfd 2482c82a c27e8f5c cbdccc6a 7fcf0222 aa1ff21a 90294621 20cd8cd6 6c96797f 9c18fc18 8f1df778 e95e96da 0aa257e7 560993e1 602c7983 6e2a11cc 4d44afda 0ed4fa52 35a2bdd3 6abd62b6 bdca1656 ab1b1946 1c10af18 c6a9d0fc 4c473992 638f9747
Second part of signature s	1ecf7056 cac6b0d4 a951f8b6 9e9c191f 930a101e f3f891ff d1636615 b2444590 c1a0e3ee af8f701d 4a796761 d64fcda2 7622fe9f f0645eba 617e9747 2bafc0bf f487efd0 2d2ca4c1 7705a1e6 0c68c6a9 fadd5ca5 43988d5f a338f5e1 5bb59edf 41ce6ecc 2c8832f2 a0565e81 f1696845 2f99ae59 ad24c5d8 bb70a148 9f65a37d

F.1.3 Example with Dedicated Hash-Function 1 (otherwise known as RIPEMD-160) of ISO/IEC 10118-3

Length of hash-token	21 octets
Recoverable length L_{rec}	00000000 0000006a
Non-recoverable length L_{clr}	00000000 0000008e
Hash-code	525d1604 e8a2a6f6 054ba7a9 ffc4a18e bab0fe2b
Hash-function identifier	31
Data input d	525d16 04e8a2a6 f6054ba7 a9ffc4a1 8ebab0fe 2b314142 43444546 4748494a 4b4c4d4e 4f505152 53545556 5758595a 61626364 65666768 696a6b6c 6d6e6f70 71727374 75767778

797a3031 32333435 36373839 41424344 45464748 494a4b4c 4d4e4f50
51525354 55565758 595a6162 63646566 6768696a 6b6c6d6e 6f707172

First part of signature r	0f0e7821 bfdc63c8 f52f2400 2635a8cc e4cfb00f d572102f 88909cfd 2482c82a c27e8f5c cbdccc6a 7fcf0222 aa1ff21a 90294621 20cd8cd6 6c96797f 9c18fc18 8f1df778 e95e96da 0aa257e7 560993e1 602c7983 6e2a11cc 4d44afda 0ed4fa52 35a2bdd3 6abd62b6 bdca1656 ab1b1946 1c10af18 c6a9d0fc 4c473992 638f9747
Second part of signature s	3e1bf266 a2fe5226 79192ef9 14e4f648 3a89a3c4 87243e86 beecfae9 dabbec98 eaff37d0 b3eaab2c 2308becc b3681577 9de664bb 4547c06c 8e456be2 24488268 649c30e2 ffb25460 86745066 20e5c853 d6194981 607a2386 be38f463 dd820d10 32771638 7c874364 1ab116eb 00421592 e70b7281 2746acfc 19b601fc 6de5a89d

F.1.4 Example with Dedicated Hash-Function 2 (otherwise known as RIPEMD-128) of ISO/IEC 10118-3

Length of hash-token	17 octets
Recoverable length L_{rec}	00000000 0000006e
Non-recoverable length L_{clr}	00000000 0000008a
Hash-code	ab8fd266 ddddbddc 48d117ea f0968b0c
Hash-function identifier	32
Data input d	ab8fd2 66ddddbd dc48d117 eaf0968b 0c324142 43444546 4748494a 4b4c4d4e 4f505152 53545556 5758595a 61626364 65666768 696a6b6c 6d6e6f70 71727374 75767778 797a3031 32333435 36373839 41424344 45464748 494a4b4c 4d4e4f50 51525354 55565758 595a6162 63646566 6768696a 6b6c6d6e 6f707172 73747576
First part of signature r	0f67aade 21d19edf db72a970 67267ab6 62474053 ed851433 8c94a101 2886cc2e c6829360 cfe0d06e 83d30626 b429fc24 942d4a25 24d190da 709a7d83 a01d001c 9321fb7c ed624f92 c35b5beb 5a0d97e5 6b37848e 722e15d0 5148b3de 12d8fe56 39a6c1d7 6ec166ba c1ce2060 b5251d4a 2014b31c caadd500 504b3d96 67939b4b
Second part of signature s	64dc5bce 568cb0be 22ea47f7 d848a5ef fc34fdea 0f11ed67 ee24753f 655e72fa c0d12fed da5f0c13 9c9d1544 8cce2297 6a2b0fb0 00055fd8 4e0d38b9 86fde806 fc74e1d4 ddd8144d dd5530a1 66fd03aa 11003478 06e5678f 7dd9927a 5834c0d2 cdffb15c 14dec608 bb6eac7c 15a3c6c7 05de2a82 4b5a3e9f f4b26171 9b8daf16

F.1.5 Example with total recovery (RIPEMD-128)

P	1 5654b2a4 8af38b0b 45b10960 41a7f552 4a97a065 fff0bb31 94cae13e 38c2969e 527dc350 e5b32309 fc3342fb 741c4294 54020173 aaf8a23d 2ca4a294 27bd8c1c 6384db95 5c944e40 c321a896 b4d50969 e869d23f 49bf2489 c918c3b3 636e4907 3162512d 5ce35acc 858f70b6 daaf970e 0086bad1 2062a127 a2afeee0 5dfd9e3d
Q	1 9cafd651 31c5c9a7 d546e3f9 42577f24 220f1b07
Length of Q	161 bits

G	1 3e2cfad2 bd5e8128 c6f968df 664ab926 9d3b1ae3 a558100b 22682671 46421b71 43eb37ef 659e992a 0746c1df dc7b899e 735063d4 e4a9dcad 46f85bb5 4ffe1774 62e18fe4 43efb87f cda522bf f3097853 d5a1f723 bcde771f 903b7c0a 89974ab2 efc94b69 4590b2b1 02ed7160 f207d18b 0c748186 34118dbe c1ff775a b3a16be4
Signature key x_A	478bbe64 7cd50ef3 67ebe30f dc10c9e0 1ce37fb5
Verification key Y_A	6ce2e099 1ca9f778 571ab62d d535bbd7 260a481f 19619944 0739667f 5978cc7c 7eb25030 d3abe64a b599c1af 6414bed8 3505e2c0 8a42acf7 f2fdab50 6963399c f5b7303d 8953b565 edb0efee 6d8ae8af eeba1890 63c72571 b586092b 1fc0f9d9 d1f82cd0 6bf307bc a385dc4c 1a8cfc87 9bc622d1 135277ac 7264ebef b1fd4127
Randomizer k	5a474224 778948f7 c2aa8890 61fbb3a9 750ec2cb
Π	00 9981665a fddb49bd 99b94480 d0f314e0 9db3539e 2874acf5 59ed6376 15886470 f9a85e0f 98631dc1 516a6c4b 68c7c35d 28efcb29 c1ecd84f ed57a9ad a22d8f3f 57247312 e12c21b9 21d792be c964aed2 48b440b7 a8043ef7 fec79008 186749fd c11f4f6f fe1b734a 4a504fb3 eca68d2f 8f105a52 fe97effc 0e67fad1 14de9d02
Message to be signed	Plaintext
M	70 6c61696e 74657874
Length of truncated hash-token	10 bytes
Recoverable length L_{rec}	00000000 00000009
Non-recoverable length L_{clr}	00000000 00000000
Truncated hash-token	c42a ebdb3c89 50e9beff
Data input d	c42aeb db3c8950 e9beff70 6c61696e 74657874
First part of signature r	9af324f6 16d169d0 f1fe0dc1 7d96d4b7 e5e9f641
Second part of signature s	dc2c286e 4d5e6c1e 14551148 12eecfc5 a8f123eb

F.2 Numerical examples for ECNR

NOTE 1 The hash function is $\text{Hash}(T) = \text{RIPEMD-160}(T \,\|\, C)$, where C = 00000001 in hexadecimal.

NOTE 2 The data input is computed as the rightmost L_{dat} octets of $H \,\|\, M_{\text{rec}}$, where H is a truncated hash token of $\text{Hash}(C_{\text{rec}} \,\|\, C_{\text{clr}} \,\|\, M_{\text{rec}} \,\|\, M_{\text{clr}} \,\|\, \Pi)$. The truncated hash token is the leftmost L_{red} octets of the hash token. We used $C_{\text{rec}} = \text{I2OSP}(L_{\text{rec}}, 4)$, $C_{\text{clr}} = \text{I2OSP}(L_{\text{clr}}, 4)$.

F.2.1 Elliptic curve over a prime field

P	ffd5d55f a9934410 d3eb8bc0 4648779f 13174945

Equation of E	$y^2 \equiv x^3 + ax + b \pmod p$
A	710062dc b53dc6e4 2f8227a4 fbac2240 bd3504d4
B	4163e75b b92147d5 4e09b0f1 3822b076 a0944359
x-coordinate of G	3c1e27d7 1f992260 cf3c31c9 0d80b635 e9fd0e68
y-coordinate of G	c436efc0 041bbf09 47a304a0 05f8d43a 36763031
n (order of G)	2aa3a38f f1988b58 235241ee 59a73f46 46443245
Length of n in bits	158 bits
$L(n)$	20
L_{dat}	19
L_{red}	9
L_{rec}	10
L_{clr}	13
Signature key x_A	24a3a993 ab59b12c e7379a12 3487647e 5ec9e0ce
x-coordinate of Y_A	e564ac ae27d227 1c4af829 cface6de cc8cdce6
y-coordinate of Y_A	7bd48ce1 08ffd3cf a38177f6 83b5bcf4 fd97a4a9
k	08a8bea9 f2b40ce7 40067226 1d5c05e5 fd8ab326
$kG = (x, y)$	
x	177b7c44 ac2f7f79 96aefd27 c68d59e0 f8e01599
y	399ea116 298975bb 449d126f 6c97bddf c4e8782e
Π	02 177b7c44 ac2f7f79 96aefd27 c68d59e0 f8e01599
Message to be signed	This is a test message!
M	546869 73206973 20612074 65737420 6d657373 61676521
M_{rec}	5468 69732069 73206120
M_{clr}	74 65737420 6D657373 61676521
Hash input	0000000a 0000000d 54686973 20697320 61207465
	7374206d 65737361 67652102 177b7c44 ac2f7f79
	96aefd27 c68d59e0 f8e01599
Truncated hash-token H	64 1e6fe77e b1b9cca9
Data input $d = H \Vert M_{\text{rec}}$	641e6f e77eb1b9 cca95468 69732069 73206120
First part of signature r	1833eff5 4087a911 bb7d3a63 fc2982ff 20ce1b7d

Second part of signature s	155d498e 35855ab5 04b9adda 0315ca77 4b171e61

F.2.2 Elliptic curve over an extension field $\mathrm{GF}(2^m)$

Galois field $\mathrm{GF}(2^{185})$ with the polynomial $x^{185} + x^{69} + 1$.

Equation of E	$y^2 + xy = x^3 + ax^2 + b$
A	07 2546b543 5234a422 e0789675 f432c894 35de5242
B	00 c9517d06 d5240d3c ff38c74b 20b6cd4d 6f9dd4d9
x-coordinate of G	07 af699895 46103d79 329fcc3d 74880f33 bbe803cb
y-coordinate of G	01 ec23211b 5966adea 1d3f87f7 ea5848ae f0b7ca9f
N	04 00000000 00000000 0001e60f c8821cc7 4daeafc1
Length of n	163 bits
$L(n)$	21
L_{dat}	20
L_{red}	10
L_{rec}	10
L_{clr}	13
x_A	03 d648bcb2 e4d5d151 656c8477 4ed016ba 292a5a38
x-coordinate of Y_A	07 01b9786f d72171da a883f34c 44deeace a10b8d02
y-coordinate of Y_A	00 0149bdc1 54c6ab7f 4e5b4a4a 57d528d7 65d7f8ea
k	02 887ac572 8a839081 8b535fcb f04e827b 0f8b543c
$kG=(x,y)$	
x	00 eeddbbcf 22652313 c3484118 5d3ebb53 8c453aee
y	03 7df0f68a c78cd813 0a6ffeda 5ba85ff1 14e93ec7
Π	0200 eeddbbcf 22652313 c3484118 5d3ebb53 8c453aee
Message to be signed	This is a test message!
M	546869 73206973 20612074 65737420 6D657373 61676521
M_{rec}	5468 69732069 73206120
M_{clr}	74 65737420 6D657373 61676521
Hash input	00 00000a00 00000d54 68697320
	69732061 20746573 74206d65 73736167 65210200

	eeddbbCf 22652313 c3484118 5d3ebb53 8c453aee
Truncated hash-token H	d8f3 55fde3c6 1cb29bc0
Data input $d = H \parallel M_{\text{rec}}$	d8f355fd e3c61cb2 9bc05468 69732069 73206120
First part of signature r	01 c7d111cd 062b3fc6 5e158d9c 85a37816 280dbb8e
Second part of signature s	01 0fe6b789 d7ef86bc 5ed726ca 0fc7c96c f6b4faa3

F.2.3 Elliptic curve over an extension field $\mathrm{GF}(p^m)$

p	fffffffb
m	5
Irreducible polynomial	$X^5 - 2$
Equation of E	$y^2 = x^3 + ax + b$
a	00000000 00000000 00000000 00000000 00000000 00000000 00000001
b	00000000 00000000 00000000 00000000 00000000 00000001 00000106
x-coordinate of G	fcdee3ee eb6a9d0c 821c8b46 d27937bc 0fbac840
y-coordinate of G	3c329e0d 7a5fb6e4 048a69c1 12f8cb35 dffb7ccc
n	ffffffe7 000000f9 fffe3308 f697c1d6 d7de35cf
Length of n	160 bits
$L(n)$	20
x_A	d648bcb2 e4d5d151 656c8477 4ed016ba 292a5a38
x-coordinate of Y_A	be00180e c77feb6e a550dbf6 a6d5ccce 8b1f7cf6
y-coordinate of Y_A	13ad8b66 c59205f7 71112f36 effa0650 72487bef
k	887ac572 8a839081 8b535fcb f04e827b 0f8b543c
$kG=(x,y)$	
x	c9c83609 b667081f 09d4f822 325daa91 01e06c84
y	4e95c220 783a466f 2d2f12aa 6ee07c60 928d2594
Π	02 c9c835f9 f2c2cfd6 951b2642 2531d251 8d5547d7
Message to be signed	This is a test message!
M	546869 73206973 20612074 65737420 6d657373 61676521
M_{rec}	5468 69732069 73206120

M_{clr}	74 65737420 6d657373 61676521
Hash input	0000000a 0000000d 54686973
	20697320 61207465 7374206d 65737361 67652102
	c9c835f9 f2c2cfd6 951b2642 2531d251 8d5547d7
Truncated hash-token H	547a d9d64b5e 9b62e920
Date input $d = H \parallel M_{\text{rec}}$	7ad9d6 4b5e9b62 e9205468 69732069 73206120
First part of signature r	ca431002 3e216945 7e3f1498 a1756f0d 50b93d59
Second part of signature s	951cd069 e020eb4d 3da1c3dc e316819c 260c8d36

F.3 Numerical examples for ECMR

F.3.1 Elliptic curve over a prime field

NOTE (1) Truncated hash token *h* is first *L* octets of the Dedicated Hash-Function 3 (otherwise known as SHA-1) from ISO/IEC 10118-3 output of $\Pi \parallel M$.

(2)The function Mask is SHA-1.

p	ffffffff ffffffff ffffffff ffffffff ffff7c67
Equation of E	$y^2 \equiv x^3 + ax + b \pmod p$
A	ffffffff ffffffff ffffffff ffffffff ffff7c64
B	26c1d102 82415e10 a4995e19 80b59224 d7120957
Number of points on E	ffffffff ffffffff ffffc748 a4eea1b0 dc8744b9
x-coordinate of G	1
y-coordinate of G	22e0d7c6 1eb0627b 334456c7 a50b77fd a9007da6
N	ffffffff ffffffff ffffc748 a4eea1b0 dc8744b9
Length of n	160 bits
Signature key x_A	ddd259e3 d30a77ab c31cdf29 9a0e6cff 7d78f869
x-coordinate of Y_A	6de7e135 f5b2ad0c e33492fa 61ed55b0 a00be7ba
y-coordinate of Y_A	79473d9c ea21791a 391d536c 99ebfb13 4b94c3cc
Randomizer k	4b13079f a8f2992e 5bcdb38d 6895a31b 91d822c2
x-coordinate of kG	b4f8c602 dec23b19 358271b8 fe868ad4 a7f7fa9f
y-coordinate of kG	691b933a c8e59096 63364135 f5015637 f337f424
$\Pi = (\text{Mask}(\text{EC2OSP}_E(kG))$	e5a0d1f5 85a1cf2c 431a8c61 b78f316c 80d8928f

Message to be signed	TestVector
$M(=M_{rec})$ (M_{clr} is empty)	5465 73745665 63746f72
Length $L(=L_1)$ of truncated hash-token	10 octets
Recoverable length L_{rec}	10 octets
Non-recoverable length L_{clr}	0 octet
Truncated hash-token h	3b16 b61a504b 21855dfc
Data input $d = h \parallel M$	3b16b61a 504b2185 5dfc5465 73745665 63746f72
First part of signature r	deb667ef d5eaeea9 1ee6d804 c4fb6709 e3acfdfd
Second part of signature s	1f5d610b b13e61c9 03a24f8f 1af14c0a 122cc560

F.3.2 Elliptic curve over an extension field $\mathrm{GF}(2^m)$

Galois field $\mathrm{GF}(2^{163})$ with the polynomial $x^{163} + x^7 + x^6 + x^3 + 1$.

NOTE (1)This is a standard polynomial basis implementation.

(2)The function Mask is MGF1 based on SHA-1.

Equation of E	$y^2 + xy = x^3 + ax^2 + b$
a	1
b	2 0a601907 b8c953ca 1481eb10 512f7874 4a3205fd
Number of points on E	8 00000000 00000000 000525fc efce1825 48469866
x-coordinate of G	3 f0eba162 86a2d57e a0991168 d4994637 e8343e36
y-coordinate of G	0 d51fbc6c 71a0094f a2cdd545 b11c5c0c 797324f1
n	4 00000000 00000000 000292fe 77e70c12 a4234c33
Length of n	163 bits
x_A	2 ddd259e3 d30a77ab c31cdf29 9a0e6cff 7d78f869
x-coordinate of Y_A	6 a15faa2f 38cabcbc 48113b58 6c5148a7 f80c424c
y-coordinate of Y_A	3 302077a6 3ea741d4 ecf200cf 68cd272f b21eefdc
Randomizer k	3 97e49b66 4b13079f a8f2992e 5bcdb38d 6895a31b
x-coordinate of kG	7 3b811311 c037c110 38350437 95543abd 067af556
y-coordinate of kG	6 0f7b188b 0ad4345b 910c0a1f 7b301c31 f9f8d9e1
$\Pi = \mathrm{Mask}(\mathrm{EC2OSP}_E(kG))$	e7 acd53a64 16db34c1 788b2011 edaa0db7 9bbd9a21

Message to be signed	TestVector
M(=M_{rec}) (*M_{clr}* is empty)	5465 73745665 63746f72
Length $L(=L_1)$ of truncated hash-token	11 octets
Recoverable length L_{rec}	10 octets
Non-recoverable length L_{clr}	0 octet
Truncated hash token h	c76dd5 cc49fa1a bc0aabb4
Data input $d = h \parallel M$	c7 6dd5cc49 fa1abc0a abb45465 73745665 63746f72
First part of signature r	20 c100f62d ecc188cb d33f7474 9ede5bd2 f8c9f553
Second part of signature s	0 2dd0bfcb f8745141 33cdf701 fe774ae3 ff2d7d16

F.3.3 Elliptic curve over an extension field $\mathrm{GF}(p^m)$

NOTE (1) An element τ in $\mathrm{GF}(p^m)$ is defined as $t_4x^4 + t_3x^3 + t_2x^2 + t_1x + t_0$ and denoted as $t_4\ t_3\ t_2\ t_1\ t_0$.

(2) The function Mask is SHA-1.

p	ffffff47
m	5
Irreducible polynomial	$x^5 - 2$
Equation of E	$y^2 \equiv x^3 + ax + b \pmod{p}$
a	00000000 00000000 00000000 00000000 ffffff44
b	39cd7fda f41a7fb5 488651a5 e362f27f b449e900
Number of points on E	fffffc63 000538e9 fc3bbe32 da01dc69 c2516d77
x-coordinate of G	00000000 00000000 00000000 00000000 00000002
y-coordinate of G	8a45f6c7 82f3c45e e2716ce9 26573f3f c5105399
n	fffffc63 000538e9 fc3bbe32 da01dc69 c2516d77
Length of n	160 bits
x_A	7b5f8464 0e65495c 87e807aa 22b446fb 34e77471
x-coordinate of Y_A	a39e766 99f8ec98 26c87346 6dd50ba2 94116c31
y-coordinate of Y_A	9b2ef2cf 22061787 54b154b9 acc2a731 359b675b
Randomizer k	8819bc2c 9ef7fdfc 2c348697 cadbc2be 77349b87
x-coordinate of kG	c4e47f64 de0d2859 33af7a91 c5252d1f 671b20a2

y-coordinate of kG	ca71997e 285b76f9 292af138 c4267642 eb1458b9
$\Pi = (\mathrm{Mask}(\mathrm{EC2OSP}_E(kG))$	d1fdf8a3 d5bcb759 7cc5b859 1c2d2269 2e0a7cd2
Message to be signed	TestVector
$M(=M_{\mathrm{rec}})$ (M_{clr} is empty)	5465 73745665 63746f72
Length $L(=L_1)$ of truncated hash-token	10 octets
Recoverable length L_{rec}	10 octets
Non-recoverable length L_{clr}	0 octets
Truncated hash token h	55d7 dd9166fd 83eca94f
Data input $d = h \parallel M$	55d7dd91 66fd83ec a94f5465 73745665 63746f72
First part of signature r	842a2532 b34134b5 d58aec3c 6f59740c 4d7e13a0
Second part of signature s	8dd81109 707daa15 df465d58 008073fe 573f4ca2

F.4 Numerical examples for ECAO

NOTE 1 In the numerical examples described in Clauses F.4.1 through F.4.6,

— Hash_1 uses L_{red} leftmost octets of the Dedicated Hash-Function 4 (otherwise known as SHA256) from ISO/IEC 10118-3,

— Hash_2 uses $(L_F + 1 - L_{\mathrm{red}})$ leftmost octets of SHA256,

— MGF is constructed from MGF1 with SHA256 as the underlying hash-function,

— $K = L(n)$, and

— the Key Generation Scheme I (as described in Clause 7.3) is used, which implies that $P = G$ and $Q = Y_A$.

NOTE 2 In the numerical examples described in Clause F.4.2, the domain parameter, user keys and the randomizer are the same as those described in Clause F.4.1.

NOTE 3 In the numerical examples described in Clause F.4.4, the domain parameter, user keys and the randomizer are the same as those described in Clause F.4.3.

NOTE 4 In the numerical examples described in Clause F.4.6, the domain parameter, user keys and the randomizer are the same as those described in Clause F.4.5.

F.4.1 Elliptic curve over a prime field $\mathrm{GF}(p)$ (total message recovery)

p	`ffffffff ffffffff ffffffff fffffffe ffffffff ffffffff`
L_F	24 octets
E	$y^2 = x^3 + ax + b$
a	`ffffffff ffffffff ffffffff fffffffe ffffffff fffffffc`
b	`64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1`
x-coordinate of P	`188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012`
y-coordinate of P	`07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811`
n	`ffffffff ffffffff ffffffff 99def836 146bc9b1 b4d22831`
$L(n)$	24 octets
L_{red}	12 octets
$L_{\max} = L_F - L_{\mathrm{red}}$	12 octets
x_A	`a662ee37 61adf2bb ca1c1695 9b1de2a4 3d4cd1bf 10937f21`
x-coordinate of Q	`b6b54a54 a240cfe5 68a339f2 55739317 61f6094f 6eabad7b`
y-coordinate of Q	`96c96e37 09b7d3d5 d34579fc b3f9b11f 390d29f3 27b3eeb9`
k	`722c12ad afa68860 758d37b1 f23f5dad c292bdcc d6373358`
x-coordinate of R	`3a01544a f34294ff 693cd261 56eb921c d883e4ab 58dc922e`
y-coordinate of R	`cdbd2710 8c16fbc2 54aaa2e9 d6fcdad0 130e487c 04fc4319`
Π	`03 3a01544a f34294ff 693cd261 56eb921c d883e4ab 58dc922e`
M	`70 6c61696e 74657874` (ASCII encoding of “plaintext”, 9 octets)
M_{rec}	`70 6c61696e 74657874` (9 octets)
$\tilde{M}_{\mathrm{rec}}$	`00 00000170 6c61696e 74657874`
$\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}})$	`8561b655 efea344e 0d080ce2`
$\mathrm{Hash}_2(\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}}))$	`0b 96756661 7c9fb00b de68fe1d`
d	`85 61b655ef ea344e0d 080ce20b 96756711 10fed965 aa0d8669`
r	`86 5bb701a5 1976daf2 6130306a c09ef50d c87d3dce f2d11447`
M_{clr}	(null octet)
u	`2ddd605e 5d532b78 003e3175 574fbc25 9aea57e2` `ad2548cf f8f67863 9a73c6b6 2393be70 dd3e448e b1c2b672 f4d03129`
t	`e155f602 4de3a6b5 22f4ac81 866eb9bc ada0b108 bab96c8b`
s	`8a671200 94491474 3b0c47f8 40759e60 bfbfa2dd e6538f3c`

F.4.2 Elliptic curve over a prime field $\mathrm{GF}(p)$ (partial message recovery)

M	546869 73206973 20612074 65737420 6d657373 6167652e (ASCII encoding of "This is a test message.", 23 octets)
M_{rec}	54686973 20697320 61207465 (12 octets)
$\tilde{M}_{\mathrm{rec}}$	01 54686973 20697320 61207465
$\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}})$	1f977b66 6a2fab85 d9e126af
$\mathrm{Hash}_2(\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}}))$	e6 ba4831b4 dd1e3707 4ea32f12
d	1f 977b666a 2fab85d9 e126afe7 ee2058c7 fd774427 2f835b77
r	1c ad7a3220 dce91126 881a7d86 b8cbcadb 25f4a08c 775fc959
M_{clr}	737420 6d657373 6167652e (11 octets)
u	86541ea8 0371f4ea b2253742 4dd4cbac a3a2365b 0221cccc b5228949 eada5763 8fc78e62 094e7059 2919e64e 9fcce43a
t	38b5bd83 50710f7f 0d680ab2 238317a2 cf7604f5 c4ca3e16
s	c6daa1eb e8b23a89 593d16f2 f62ed760 9b7822be 950dc612

F.4.3 Elliptic curve over a binary field $\mathrm{GF}(2^m)$ (total message recovery)

m	193
Irreducible polynomial	$X^{193} + X^{15} + 1$
L_F	25 octets
E	$y^2 + xy = x^3 + ax^2 + b$
a	00 17858feb 7a989751 69e171f7 7b4087de 098ac8a9 11df7b01
b	00 fdfb49bf e6c3a89f acadaa7a 1e5bbc7c c1c2e5d8 31478814
x-coordinate of P	01 f481bc5f 0ff84a74 ad6cdf6f def4bf61 79625372 d8c0c5e1
y-coordinate of P	00 25e399f2 903712cc f3ea9e3a 1ad17fb0 b3201b6a f7ce1b05
n	01 00000000 00000000 00000000 c7f34a77 8f443acc 920eba49
$L(n)$	25 octets
L_{red}	12 octets
$L_{\mathrm{max}} = L_F - L_{\mathrm{red}}$	13 octets
x_A	00 dda332f9 340562e1 d2aec168 249b5696 ee39d0ed 4d03760f
x-coordinate of Q	01 52907843 50956c4e fd9ba320 ab267cca 7cc1b4a3 94eba91d
y-coordinate of Q	01 0cdc62d5 0ccff338 2b69a546 48e778a7 a90c077d ce8ef4ba
k	00 8afb928e 1ae15cc3 2517229e ffc84c9e 0d4040b5 10d8b509

x-coordinate of R	01 3bc6bb62 c2b49945 fdb2c85a 095b9ca2 b70ca8cf c46d9a6f
y-coordinate of R	00 ed7bec1e c8a60cd2 35446995 ed27cad1 17cac0dc 2a38e6ab
Π	0301 3bc6bb62 c2b49945 fdb2c85a 095b9ca2 b70ca8cf c46d9a6f
M	70 6c61696e 74657874 (ASCII encoding of “plaintext”, 9 octets)
M_{rec}	70 6c61696e 74657874 (9 octets)
$\tilde{M}_{\mathrm{rec}}$	00 00000170 6c61696e 74657874
$\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}})$	f9d89601 8f531a6c 20555d7e
$\mathrm{Hash}_2(\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}}))$	dd54 07b02957 06b3902e 3bb4c0e5
d	f9d8 96018f53 1a6c2055 5d7edd54 07b02827 6ad2f940 4fd1b891
r	fad9 adc73431 d8d8b910 a0cc150e 0eebb485 ddde518f 8bbc22fe
M_{clr}	(null octet)
u	f6c4 70030564 5647caf8 6b9a672e d7955b64 5d95b106 37d932e0 0549a825 5b1a903a 5c8f65de 50519044 961b2a32 c44f428a
t	00 ea8d687c 4aa61ae4 2f3431a6 a5179174 e95cf384 1e2b6f52
s	00 af5fd7c1 ffc008eb 4ca5feb5 00dbac4d 2e93c70e c47e83ad

F.4.4 Elliptic curve over a binary field $\mathrm{GF}(2^m)$ (partial message recovery)

M	546869 73206973 20612074 65737420 6d657373 6167652e (ASCII encoding of “This is a test message.”, 23 octets)
M_{rec}	54 68697320 69732061 20746573 (13 octets)
$\tilde{M}_{\mathrm{rec}}$	0154 68697320 69732061 20746573
$\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}})$	4648e59c d08527c8 2a6e28db
$\mathrm{Hash}_2(\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}}))$	2a1e 30db454f c5be212f 67a70950
d	4648 e59cd085 27c82a6e 28db2b4a 58b2366f accd014e 47d36c23
r	4549 de5a6be7 e57cb32b d569e310 51e9aacd 1bc1a981 83bef64c
M_{clr}	7420 6d657373 6167652e (10 octets)
u	fddf 20c33b27 18732839 bd42b5c6 8aa40b05 eded453e 1fe50281 832ca158 ab8deadf 12f6de11 de093779 b074df68 cb7a9f82
t	00 e2b6e0e9 3a691fb4 6ee29ccf bf4c842e 14ad1293 cc374906
s	00 6541adb1 0af6e8e5 1c51f531 d6f80857 f4bc3b48 f91e6c55

F.4.5 Elliptic curve over an OEF $\mathrm{GF}(p^m)$ (total message recovery)

p	ffffffef
m	7
Irreducible polynomial	$X^7 - 2$
L_F	28 octets
E	$y^2 = x^3 + ax + b$
a	ffffffec
b	ffffffa7
x-coordinate of P	53978e2d df1be9bc 5c5f449d f8ff45bb 092ce058 480c97a6 54fcfd5c
y-coordinate of P	0f67ad23 05f57ca9 cd94b85e c01312b0 1aafdc3f 347847e2 fa91f51d
n	01 0001e073 868a3c36 97a011db f31aac14 2fd06a5c edfe9a6d
$L(n)$	25 octets
L_{red}	12 octets
$L_{\max} = L_F - L_{\mathrm{red}}$	16 octets
x_A	00 9117ce9a 7f34035e a6eea5ef dcca4123 c432f75b bb1410c9
x-coordinate of Q	219fa45c 0b8270f6 a45ab7fd 9435375f 81f05f01 bbb29517 a30dd0e4
y-coordinate of Q	b1170663 f0a39ee4 a8a61faf 30c2bf31 2736ec4c 2e5da476 5de81bd1
k	00 8def996f 02c183bf 8911651b 7f1680c5 87d11403 4cb3cac0
x-coordinate of R	dbfa4a6a 406dd97a a735faec 57974884 42591243 79d43192 83fd8db4
y-coordinate of R	2d2ef1db 9ea5ad49 3deb0b51 027e6404 a2fe62cd 622db96d 439d4cb3
Π	03 dbfa4a6a 406dd97a a735faec 57974884 42591243 79d43192 83fd8db4
M	70 6c61696e 74657874 (ASCII encoding of “plaintext”, 9 octets)
M_{rec}	70 6c61696e 74657874 (9 octets)
$\tilde{M}_{\mathrm{rec}}$	00 00000000 00000170 6c61696e 74657874
$\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}})$	36466534 d29b6ede 80183942
$\mathrm{Hash}_2(\mathrm{Hash}_1(\tilde{M}_{\mathrm{rec}}))$	7a a5ddf580 d77b4415 6159ef60 a5f41457
d	36 466534d2 9b6ede80 1839427a a5ddf580 d77b4565 0d38860e d1916c23
r	35 9d9f7eb8 db0307fa bf0cb896 f24abd04 95225726 74ecb79c 526ce197

M_{clr}	(null octet)
u	`ca04 982fdafc baae9f0e e309e4d7 6235c945 d2daf992` `b220821b 3e99066f d4da4f4a cb656628 1e1423a0 2a9bc69e adb1f2c9`
t	`00 66db306f fccf5a69 075c6cc7 1d6d60ab 02edea7a 14cdfe85`
s	`00 14e79dc1 4987a74d f3ffa270 7bacd9e7 09175d84 c24d144d`

F.4.6 Elliptic curve over an OEF $\mathrm{GF}(p^m)$ (partial message recovery)

M	`546869 73206973 20612074 65737420 6d657373 6167652e` (ASCII encoding of “This is a test message.”, 23 octets)
M_{rec}	`54686973 20697320 61207465 7374206d` (16 octets)
$\tilde{M}_{\text{rec}}$	`01 54686973 20697320 61207465 7374206d`
$\mathrm{Hash}_1(\tilde{M}_{\text{rec}})$	`c1caefe5 b6b091ae ca320d45`
$\mathrm{Hash}_2(\mathrm{Hash}_1(\tilde{M}_{\text{rec}}))$	`8a e0b350ae 3b36ca13 b4dae599 abfe34e6`
d	`c1` `caefe5b6 b091aeca 320d458b b4db39dd 1b5fb933 d5fa91fc d88a148b`
r	`c2` `1115afdc f0fc77b0 9538bf67 e34c7159 5906ab70 ac2ea06e 5b77993f`
M_{clr}	`657373 6167652e` (7 octets)
u	`12fa 558ef281 30dc42d0 29252025 a3d75a32 53db6407` `4cce7e7d aa53e685 233e058b 661a2ad6 c87569ac dae34840 874f81df`
t	`00 a4a64763 8c7f0580 360420d3 a571ec5a de5f80cb 026af1c8`
s	`00 6bbbedfc babe19d7 fbe7f690 be76c76c ef8d9e6a ec9ce845`

F.5 Numerical examples for ECPV

NOTE In the numerical examples described in Clauses F.5.1 and F.5.2,

— Hash uses the Dedicated Hash-Function 3 (otherwise known as SHA1) from ISO/IEC 10118-3,

— $\mathrm{KDF}(x)$ is defined to be $\mathrm{MGF2}(x, 18)$ with SHA1 as the underlying hash-function, and

— the symmetric cipher Sym uses the exclusive-or ($\oplus$) operation.

F.5.1 Elliptic curve over a prime field $\mathrm{GF}(p)$

p	ffffffff ffffffff ffffffff fffffffe ffffac73
Equation of E	$y^2 \equiv x^3 + ax + b \pmod p$
a	00000000 00000000 00000000 00000000 00000000
b	00000000 00000000 00000000 00000000 00000007
Number of points on E	01 00000000 00000000 0001b8fa 16dfab9a ca16b6b3
x-coordinate of G	3b4c382c e37aa192 a4019e76 3036f4f5 dd4d7ebb
y-coordinate of G	938cf935 318fdced 6bc28286 531733c3 f03c4fee
n	01 00000000 00000000 0001b8fa 16dfab9a ca16b6b3
Length of n in bits	161 bits
$L(n)$	21
Signature key x_A	00 e6a080e0 b2a7a850 ba71d26c 9606669a 4b4a6c18
x-coordinate of Y_A	8f5788a5 c97ac053 984045f4 c9ff325d d60065ae
y-coordinate of Y_A	a5329d2a 721b5787 9c215323 37211f64 23e577da
M	Test User 1
M_{rec}	13 0b546573 74205573 65722031 (M_{rec} is the DER encoding of the *PrintableString* value “Test User 1”.)
L_{rec}	13
M_{clr}	fa 2b0cbe77 (M_{clr} is a random nonce which has no redundancy and will be in the clear.)
L_{red}	5
$\tilde{C}_{\text{red}}$	05 05050505 (Added redundancy of 40 bits; total redundancy is over 80 bits.)
$d = \tilde{C}_{\text{red}} \parallel M_{\text{rec}}$	0505 05050513 0b546573 74205573 65722031
Randomizer k	00 d8a0abc5 b7a4029a c232cbcd a16819e1 b715f9f4
$x_0 = x$-coordinate of R	1af46ec4 e95daede 056bfa3b 370075f6 3cb2c34f
y-coordinate of R	c324f1ba 5324383c 371278d5 f3fe4fe2 9373702c
$\Pi = \mathrm{KDF}(x_0)$	20b9 f76f3b33 6a805e02 92ed0b71 c9aaa767
$r = \mathrm{Sym}(d, \Pi) = d \oplus \Pi$	25bc f26a3e20 61d43b71 e6cd5e02 acd88756
$u = \mathrm{Hash}(r \parallel M_{\text{clr}})$	1b5361a3 9bc7ca45 6bbb6f2b f80e4e31 a01b4a1b
$s = (k - x_A t) \bmod n$ (where $t = \mathrm{OS2IP}(u)$)	735 171 855 371 469 914 412 919 293 672 210 175 623 097 579 946 (digit)

F.5.2 Elliptic curve over an extension field $\mathrm{GF}(2^m)$

m	163
Extension polynomial	$X^{163}+X^7+X^6+X^3+1$
Basis representation	A standard polynomial basis implementation
Equation of E	$y^2+xy=x^3+ax^2+b$
a	`00 00000000 00000000 00000000 00000000 00000001`
b	`00 00000000 00000000 00000000 00000000 00000001`
Number of points on E	`08 00000000 00000000 00040211 45C1981b 33f14bde`
x-coordinate of G	`02 fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8`
y-coordinate of G	`02 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9`
n	`04 00000000 00000000 00020108 a2e0cc0d 99f8a5ef`
Length of n in bits	163 bits
$L(n)$	21
x_A	`03 a41434aa 99c2ef40 c8495b2e d9739cb2 155a1e0d`
x-coordinate of Y_A	`03 7d529fa3 7e42195f 10111127 ffb2bb38 644806bc`
y-coordinate of Y_A	`04 47026eee 8b34157f 3eb51be5 185d2be0 249ed776`
M	Test User 1
M_{rec}	`13 0b546573 74205573 65722031` (M_{rec} is the DER encoding of the *PrintableString* value “Test User 1”.)
L_{rec}	13
M_{clr}	`fa 2b0cbe77` (M_{clr} is a random nonce which has no redundancy and will be in the clear.)
L_{red}	5
$\tilde{C}_{\text{red}}$	`05 05050505` (Added redundancy of 40 bits, total redundancy is over 80 bits.)
$d=\tilde{C}_{\text{red}} \parallel M_{\text{rec}}$	`0505 05050513 0b546573 74205573 65722031`
Randomizer k	`a40b301c c315c257 d51d4422 34f5aff8 189d2b6c`
$x_0 = x$-coordinate of R	`04 994d2c41 aa30e529 52b0a94e c6511328 c502da9b`
y-coordinate of R	`03 1fc936d7 3163b858 bbc5326d 77c19839 46405264`
$\Pi=\mathrm{KDF}(x_0)$	`d5bd 8bd7309d 020d5946 1367e723 a3b63d79`
$r=\mathrm{Sym}(d,\Pi)=d\oplus\Pi$	`d0b8 8ed2358e 09593c35 6747b250 c6c41d48`
$u=\mathrm{Hash}(r \parallel M_{\text{clr}})$	`db9b5ebe 6b7b9690 54f9aecb 52b54a19 df4495ae`
$s=(k-x_A t) \bmod n$ (where $t=\mathrm{OS2IP}(u)$)	2 748 261 816 569 194 283 991 299 038 913 308 940 368 752 275 658 (digit)

F.6 Numerical examples for ECKNR

NOTE 1 This is a standard polynomial basis implementation.

NOTE 2 MGF2 uses RIPEMD-160 as its underlying hash-function.

NOTE 3 The method of producing the data input is as described in Annex D, with RIPEMD-160 as Hash.

F.6.1 Elliptic curve over a prime field

p	`ffffffff ffffffff 0a13a2a0 a085053b 49a92b05`
Equation of E	$y^2 \equiv x^3 + ax + b \pmod p$
a	`ffffffff ffffffff 0a13a2a0 a085053b 49a92b02`
b	`809dc828 d7ec47f1 d1b20800 62a9d350 c3e7b230`
x-coordinate of G	`ae316775 14f76709 513c8442 4165d440 0bb7d699`
y-coordinate of G	`71fa37b7 27cbd843 c800d474 1448267a 8fdd047e`
n	`ffffffff ffffffff 0a15341c 63139e6c 9e868967`
Length of n in bits	160
$L(n)$	20
L_{dat}	20
L_{red}	10
L_{rec}	10
L_{clr}	13
Signature key x_A	`d648bcb2 e4d5d151 656c8477 4ed016ba 292a5a38`
x-coordinate of Y_A	`e3daf394 34a4b15c 633a76de d8ada3de 0a70bbe1`
y-coordinate of Y_A	`2e32d49c 1b14fdc3 efa071d7 e864cf13 4b8d55af`
Cert_A	`e3daf394 34a4b15c 633a76de` `d8ada3de 0a70bbe1 2e32d49c 1b14fdc3 efa071d7 e864cf13 4b8d55af`

z_A	`e3daf394 34a4b15c`
	`633a76de d8ada3de 0a70bbe1 2e32d49c 1b14fdc3 efa071d7 e864cf13`
	`4b8d55af 00000000 00000000 00000000 00000000 00000000 00000000`

Randomizer k	`887ac572 8a839081 8b535fcb f04e827b 0f8b543c`
x-coordinate of kG	`0dc9bb8b 272b418d 2a2516da b5642d2d 41321865`
y-coordinate of kG	`8540597c 06c9bd33 7f151eb4 982f7c76 23d75fc5`
Π	`7ecdd6b7 aed42dd7 76e33cb4 43687001 39e2572f`
Message to be signed	This is a test message!
M	`546869 73206973 20612074 65737420 6d657373 61676521`

M_{rec}	5468 69732069 73206120
M_{clr}	74 65737420 6d657373 61676521
Data (hash input)	000000 00000000 0a000000 00000000 0d546869 73206973 20612074 65737420 6d657373 61676521 7ecdd6b7 aed42dd7 76e33cb4 43687001 39e2572f
Truncated hash-token h	5b72 1142ea6e 7011bbe6
Data input $d = h \parallel M_{\text{rec}}$	5b721142 ea6e7011 bbe65468 69732069 73206120
$\text{MGF}(z_A \parallel M_{\text{clr}})$	b0324c93 8ca97523 b925df2c b2be0374 6e834238
First part of signature r	958d8b66 c81328e5 7420b7f0 98a5531c 24417437
Second part of signature s	a1f82e55 b662e65b 579fe513 1c76d17b 1a71470a

F.6.2 Elliptic curve over an extension field $\text{GF}(2^m)$

m	163
Irreducible polynomial	$X^{163} + X^8 + X^2 + X + 1$
Equation of E	$y^2 + xy = x^3 + ax^2 + b$
a	07 2546b543 5234a422 e0789675 f432c894 35de5242
b	00 c9517d06 d5240d3c ff38c74b 20b6cd4d 6f9dd4d9
x-coordinate of G	07 af699895 46103d79 329fcc3d 74880f33 bbe803cb
y-coordinate of G	01 ec23211b 5966adea 1d3f87f7 ea5848ae f0b7ca9f
n	04 00000000 00000000 0001e60f c8821cc7 4daeafc1
Length of n in bits	163
$L(n)$	21
L_{dat}	21
L_{red}	10
L_{rec}	11
L_{clr}	12
x_A	03 d648bcb2 e4d5d151 656c8477 4ed016ba 292a5a38
x-coordinate of Y_A	05 85bc0f8e b4b5adf5 79dcf2c3 2c8144dc 45a38a97
y-coordinate of Y_A	04 82de581e 443872d5 6b40700d f41033f8 4ce2d205

Cert_A	0585 bc0f8eb4 b5adf579 dcf2c32c 8144dc45 a38a9704 82de581e 443872d5 6b40700d f41033f8 4ce2d205
z_A	0585bc0f 8eb4b5ad f579dcf2 c32c8144 dc45a38a 970482de 581e4438 72d56b40 700df410 33f84ce2 d2050000 00000000 00000000 00000000 00000000 00000000
Randomizer k	02 887ac572 8a839081 8b535fcb f04e827b 0f8b543c
x-coordinate of kG	06 a2338166 2db382e2 66d7b836 e3f469f5 5247ebe8
y-coordinate of kG	05 cac246d2 ff718d5f 7c5668d0 2794a7ab 7dce7210
Π	b8 4379c2b5 f49ec40c 72cb2d2d 1c79e1b5 06bf029c
Message to be signed	This is a test message!
M	546869 73206973 20612074 65737420 6d657373 61676521
M_{rec}	546869 73206973 20612074
M_{clr}	65737420 6d657373 61676521
Data (hash input)	00000000 0000000b 00000000 0000000c 54686973 20697320 61207465 7374206d 65737361 676521b8 4379c2b5 f49ec40c 72cb2d2d 1c79e1b5 06bf029c
Truncated hash token h	C333 e25a3d52 354300e9
Data input $d = h \parallel M_{\mathrm{rec}}$	c3 33e25a3d 52354300 e9546869 73206973 20612074
$\mathrm{MGF}(z_A \parallel M_{\mathrm{clr}})$	1a fa16273d fd0f4d72 3b0dc96e cfaf0cc9 f13c2f28
First part of signature r	61 8a8dbfb5 5ba4ca7e a0928c2a a0f6840f d7e20dc0
Second part of signature s	02 11b0ded9 8d24aa2b 8e66d489 930229d8 072dc2e8

F.6.3 Elliptic curve over an extension field $\mathrm{GF}(p^m)$

p	fffffffb
m	5
Irreducible polynomial	$X^5 - 2$
Equation of E	$y^2 \equiv x^3 + ax + b \pmod{p}$
a	00000000 00000000 00000000 00000000 00000001
b	00000000 00000000 00000000 00000001 00000106
x-coordinate of G	fcdee3ee eb6a9d0c 821c8b46 d27937bc 0fbac840
y-coordinate of G	3c329e0d 7a5fb6e4 048a69c1 12f8cb35 dffb7ccc

n	fffffe7 000000f9 fffe3308 f697c1d6 d7de35cf
Length of n in bits	160
$L(n)$	20
L_{dat}	20
L_{red}	10
L_{rec}	10
L_{clr}	13
x_A	d648bcb2 e4d5d151 656c8477 4ed016ba 292a5a38
x-coordinate of Y_A	1d8ea2e9 91232cfa ef256c2d 800710f3 1c2b57a6
y-coordinate of Y_A	65655866 4add4c90 b6abf35f d0c9e635 acf20dc9
Cert_A	1d8ea2e7 41fe72cf bfa93df4 f348d571 41be5813 6565585e 5ef264cf bd1641fc 92f72c58 6e29c1bd
z_A	1d8ea2e7 41fe72cf bfa93df4 f348d571 41be5813 6565585e 5ef264cf bd1641fc 92f72c58 6e29c1bd 00000000 00000000 00000000 00000000 00000000 00000000
Randomizer k	887ac572 8a839081 8b535fcb f04e827b 0f8b543c
x-coordinate of kG	fec4d254 e5ac0b25 30d7482d 7f746bc3 fa3a775a
y-coordinate of kG	3f6c5e14 ee723450 26a2750f 2e55d9e0 a4ec3e8e
Π	a17fccf4 ce99bab0 a1f76735 410f66bf 6accb85f
Message to be signed	This is a test message!
M	546869 73206973 20612074 65737420 6d657373 61676521
M_{rec}	5468 69732069 73206120
M_{clr}	74 65737420 6d657373 61676521
Data (hash input)	000000 00000000 0a000000 00000000 0d546869 73206973 20612074 65737420 6d657373 61676521 a17fccf4 ce99bab0 a1f76735 410f66bf 6accb85f
Truncated hash token h	cf53 683c2f5c 861badbf
Data input $d = h \| M_{\text{rec}}$	cf53683c 2f5c861b adbf5468 69732069 73206120
$\text{MGF}(z_A \| M_{\text{clr}})$	2eb2a254 1c2b20fc 9d58acb5 59ffd1fc 87207871
First part of signature r	409e069c fdee1c57 91109fe8 7183972a 9ecca10e
Second part of signature s	70599c96 8034ad47 b88793c2 68f8347b d24128d5

Annex G
(informative)

Summary of properties of mechanisms

In this annex, the properties of the six signature schemes giving message recovery, NR, ECNR, ECMR, ECAO, ECPV and ECKNR, are summarized. Table G.1 shows the domain parameter and user keys. Tables G.2 and G.3 show the number of operations required to generate a signature and to verify it.

Both ECAO and ECPV include operations in the process of producing the data input. Others do not include operations in the process of producing the data input.

The cost of hash-function or symmetric cipher is more than that of conversion function. However, the portion of hash computation to the total computation is negligibly small. The hash-function gives a random oracle model under which ECAO is provably secure (see reference [2]), but the conversion function does not.

Table G.1 — Summary of the six mechanisms (domain parameter and user keys)

NR	ECNR	ECMR	ECAO	ECPV	ECKNR
F, G, n x_A, Y_A P, Q L_{dat}	E, F, G, n x_A, Y_A P, Q L_{dat}	E, F, G, n x_A, Y_A P, Q Hash (or MGF)	E, F, G, n x_A, Y_A P, Q L_{red}, L_F Hash_1 Hash_2 MGF, K	E, F, G, n x_A, Y_A P, Q L_{red}, (L_{rec} or L_{clr}) Sym, L_{key} Hash KDF	E, F, G, n x_A, Y_A P, Q MGF

Table G.2 — Summary of the six mechanisms (signature generation process)

	NR	ECNR	ECMR	ECAO	ECPV	ECKNR
addition mod n	2	2	3	1	1	1
multiplication mod n	1	1	2	1	1	1
inversion mod n	0	0	1	0	0	0
scalar multiplication on an elliptic curve or exponentiation on a finite field	1	1	1	1	1	1
bitwise exclusive-OR	0	0	1	2	0	2
Hash	1	1	1 (or 0)	2	1	0
MGF or KDF	0	0	0 (or 1)	1	1	2
symmetric key cipher	0	0	0	0	1	0
I2OSP	1	1	0	1	1	0
OS2IP (mod n)	1	1	1	2	1	1
$EC2OSP_E$ or $FE2OSP_F$	1	1	1	1	1	1

Table G.3 — Summary of the six mechanisms (signature verification process)

	NR	ECNR	ECMR	ECAO	ECPV	ECKNR
addition mod n	1	1	2	0	0	0
multiplication mod n	0	0	2	0	0	0
inversion mod n	0	0	1	0	0	0
addition on an elliptic curve or multiplication on a finite field	1	1	1	1	1	1
scalar multiplication on an elliptic curve or exponentiation on a finite field	2	2	2	2	2	2
bitwise exclusive-OR	0	0	1	2	0	2
Hash	1	1	1 (or 0)	2	1	0
MGF or KDF	0	0	0 (or 1)	1	1	2
symmetric key cipher	0	0	0	0	1	0
I2OSP	1	1	0	0	1	0
OS2IP (mod n)	2	2	1	1	1	1
$EC2OSP_E$ or $FE2OSP_F$	1	1	1	1	1	1

Annex H
(informative)

Correspondence of schemes

In this annex, the correspondence among schemes in this part of ISO/IEC 9796, those in ISO/IEC 9796-3:2000 and those in ISO/IEC 15946-4:2004 is described.

This part of ISO/IEC 9796	ISO/IEC 9796-3:2000	ISO/IEC 15946-4:2004
NR (in Clause 8)	NR (in Clause 9)	—
ECNR (in Clause 9)	—	ECNR (in Clause 7)
ECMR (in Clause 10)	—	ECMR (in Clause 8)
ECAO (in Clause 11)	—	ECAO (in Clause 9)
ECPV (in Clause 12)	—	ECPV (in Clause 10)
ECKNR (in Clause 13)	—	ECKNR (in Clause 11)

Bibliography

[1] ISO/IEC 14888-1, *Information technology — Security techniques — Digital signatures with appendix — Part 1: General*

[2] M. ABE and T. OKAMOTO, "A signature scheme with message recovery as secure as discrete logarithm," *Advances in Cryptology — Asiacrypt'99*, Lecture Notes in Computer Science 1716, pp. 378-389, Springer-Verlag, 1999

[3] ANSI X9.62-1999, *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA),* 1999

[4] ANSI X9.63-1999, *Public Key Cryptography For The Financial Services Industry: Elliptic Curve Key Agreement and Transport Protocols*, 1999

[5] IEEE Std 1363-2000, *IEEE Standard Specifications for Public-Key Cryptography*

[6] C. H. LIM and P. J. LEE, "A study on the proposed Korean digital signature algorithm," *Advances in Cryptology — Asiacrypt'98*, Lecture Notes in Computer Science 1514, pp. 175-186, Springer-Verlag, 1998

[7] A. J. MENEZES, P. C. VAN OORSCHOT and S. A. VANSTONE, "*Handbook of applied cryptography*," CRC Press, 1997

[8] A. MIYAJI, "Another Countermeasure to Forgeries over Message Recovery Signature," *IEICE Trans., Fundamentals*, vol. E80-A, No.11, pp. 2192-2200, 1997

[9] K. NYBERG and R. A. RUEPPEL, "Message recovery for signature schemes based on the discrete logarithm problem," *Designs, Codes and Cryptography*, 7, pp. 61-81, 1996

[10] L. PINTSOV and S. VANSTONE, "Postal Revenue Collection in the Digital Age," *Proceedings of the Fourth International Financial Cryptography Conference*, 2000

[11] D. H. YUM, S. G. SIM and P. J. LEE, "New Signature Schemes Giving Message Recovery Based on EC-KCDSA," *Proceedings of the 12th Conference on Information Security and Cryptology (CISC)*, pp. 595-597, 2002

ICS 35.040

Price based on 69 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.

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 Group Headquarters

389 Chiswick High Road London W4 4AL UK

bsi.

...making excellence a habit.™