INTERNATIONAL STANDARD

ISO/IEC 14888-3

First edition
1998-12-15

Corrected and reprinted
1999-12-15

Information technology — Security techniques — Digital signatures with appendix —

Part 3:
Certificate-based mechanisms

Technologies de l'information — Techniques de sécurité — Signatures digitales avec appendice —

Partie 3: Mécanismes fondés sur certificat

Reference number
ISO/IEC 14888-3:1998(E)

Foreword

ISO (the International Organization for Standardization) and the 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. 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.

International Standard ISO/IEC 14888-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 14888 consists of the following parts, under the general title *Information technology — Security techniques — Digital signatures with appendix*:

— *Part 1: General*

— *Part 2: Identity-based mechanisms*

— *Part 3: Certificate-based mechanisms*

Further parts may follow.

Annexes A and B form an integral part of this part of ISO/IEC 14888. Annexes C to G are for information only.

Information technology — Security techniques — Digital signatures with appendix —

Part 3:
Certificate-based mechanisms

1 Scope

ISO/IEC 14888 specifies digital signature mechanisms with appendix for messages of arbitrary length and is applicable for providing data origin authentication, non-repudiation, and integrity of data.

This part of ISO/IEC 14888 specifies certificate-based digital signature mechanisms with appendix. In particular, this part of ISO/IEC 14888 provides 1) a general description of certificate-based digital signature mechanisms whose security is based on the difficulty of the discrete logarithm problem in the underlying commutative group (see Clause 6), 2) a general description of certificate-based digital signature mechanisms whose security is based on the difficulty of factoring (see Clause 7), and 3) a variety of normative digital signature mechanisms with appendix using certificate-based mechanisms for messages of arbitrary length (see Annex A and B).

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 14888. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 14888 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 14888-1:1998, *Information technology — Security techniques — Digital signatures with appendix — Part 1: General*.

ISO/IEC 14888-2:1999, *Information technology — Security techniques — Digital signatures with appendix — Part 2: Identity-based mechanisms*.

ISO/IEC 9796:1991, *Information technology — Security techniques — Digital signature scheme giving message recovery*.

ISO/IEC 9796-2:1997, *Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Mechanisms using a hash-function*.

ISO/IEC 10118-3:1998, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*.

ISO/IEC 10118-4:1998, *Information technology — Security techniques — Hash-functions — Part 4: Hash-functions using modular arithmetic*.

3 General

This part of ISO/IEC 14888 makes use of the definitions, symbols, legend for figures, and notation given in ISO/IEC 14888-1.

The verification of a digital signature requires the signing entity's verification key. It is thus essential for a verifier to be able to associate the correct verification key with the signing entity. For certificate-based mechanisms, this association must be provided by some certifying measure, for example, the verification key is retrieved from a certificate.

The goal of this part of ISO/IEC 14888 is to specify the following processes and functions within the general model described in ISO/IEC 14888-1:

- the process of generating keys
  - generating domain parameters
  - generating signature and verification keys

- the process of producing signatures
  - (optional) producing pre-signatures
  - preparing the message for signature

- computing witnesses
- computing the signature

- the process of verification
  - preparing message for verification
  - retrieving the witness
  - computing the verification function
  - verifying the witness

4 Definitions

For the purpose of this part of ISO/IEC 14888, the definitions of ISO/IEC 14888-1 apply. Additional definitions which are required are as follows.

4.1 Finite commutative group: A finite set J with the binary operation «∗« such that:

- For all a, b, c∈J, (a∗b) ∗ c = a ∗ (b∗c)
- There exists e∈J with e∗a = a for all a∈J
- For all a∈J there exists b∈J with b∗a = e
- For all a, b∈J, a∗b = b∗a

4.2 Order of an element in a finite commutative group: If a^0 =e, and a^{n+1}=a∗a^n (for n ≥ 0), is defined recursively, the order of a∈J is the least positive integer n such that a^n = e.

5 Symbols and notation

Throughout this part of ISO/IEC 14888 the following symbols and notations are used in addition to those given in ISO/IEC 14888-1.

E	a finite commutative group
#***E***	the cardinality of ***E***
a||b	concatenation of b to a
Q	a divisor of #***E***
G	an element of order *Q* in ***E***
gcd(*U*, *N*)	the greatest common divisor of integers *U* and *N*
T_1	first part of assignment
T_2	second part of assignment
Z_N	the set of integers *U* with $0 \le U < N$
Z_N^*	the set of integers *U* with $0 < U < N$ and gcd (*U*, *N*) = 1
⌊*a*⌋	the greatest integer equal to or less than *a*

6 Digital signature mechanisms based on discrete logarithms

6.1 Key generation process

6.1.1 Generating domain parameters

For digital signature mechanisms based on discrete logarithms, the set ***Z*** of domain parameters determines the following parameters:

- a finite commutative group ***E***
- one or more divisors *Q* of #***E***
- one or more elements *G* of order *Q* in ***E***

In the group ***E***, multiplicative notation is used. The signature mechanism will use one element *G* in ***E***. It is worthwhile to note that the particular signature mechanism chosen may place additional constraints on the choice of ***E***, *Q*, and *G*.

6.1.2 Generation of signature key and verification key

A signature key of a signing entity is a secretly generated random or pseudo-random integer *X* such that $0 < X < Q$ and gcd(*X*, *Q*) = 1. The corresponding public verification key *Y* is an element of ***E*** and is computed as

$$Y = G^X.$$

Note: It is allowed to exclude a few integers from consideration as possible X values.

In some instances, validation of domain parameters and keys may be required. However, it is outside the scope of this standard.

6.2 Signature process

In this clause the signature process for a class of signature mechanisms is described. Within this class the signature function for the mechanism to be used is specified by a permutation (A, B, C) of (S, T_1, T_2) which determines the coefficients of the signature equation.

$$AK + BX + C \equiv 0 \pmod{Q}.$$

This permutation will be specified or agreed upon when setting up the signature system.

The signature process and the formation of a signed message consists of eight stages (See Figure 1):

- producing the randomizer
- producing the pre-signature
- preparing the message for signing

- computing the witness (the first part of the signature)
- computing the assignment
- computing the second part of the signature
- constructing the appendix
- constructing the signed message

In this process, the signing entity makes use of its private signature key X, and the domain parameters ***E***, G, and Q.

6.2.1 Producing the randomizer

The signing entity generates a secret randomizer which is an integer K with $0 < K < Q$ and satisfying gcd (K, Q) = 1. The output of this stage is K, which the signing entity keeps secret.

Note: It is allowable to exclude a few integers from consideration as possible K values.

6.2.2 Producing the pre-signature

The input to this stage is the randomizer K, with which the signing entity computes

$$\Pi = G^K$$

in ***E***. The output of this stage is the pre-signature, Π.

6.2.3 Preparing the message for signing

The message is split into two parts which will be called data inputs M_1 and M_2. One of these parts may be empty and the two parts need not be distinct (See ISO/IEC 14888-1 for further details.)

6.2.4 Computing the witness (the first part of the signature)

The variables to this stage are the pre-signature Π from 6.2.2 and M_1 from 6.2.3. The values of these variables are taken as inputs to the witness function. The output of the witness function is witness R.

6.2.5 Computing the assignment

The inputs to the assignment function are the first part of the signature, which is the witness R from 6.2.4, and M_2 from 6.2.3. The output of the assignment function is assignment $T = (T_1, T_2)$ where T_1 and T_2 are integers such that

$$0 < |T_1| < Q\ ,\ 0 < |T_2| < Q.$$

6.2.6 Computing the second part of the signature

The inputs to this stage are randomizer K from 6.2.1, the signature key X, assignment $T = (T_1, T_2)$ from 6.2.5, the permutation (A, B, C) of (S, T_1, T_2) and domain parameter Q as specified in 6.1.1. The signing entity forms the signature equation

$$(AK + BX + C) \equiv 0 \pmod{Q}$$

and solves the signature equation for S, the second part of the signature, where $0<S<Q$. The pair (R, S) will be called the signature, Σ.

6.2.7 Constructing the appendix

The appendix is constructed from the signature and an optional text field, *text*, as ((R, S), *text*). The text field could include a certificate which cryptographically ties the public verification key to the identification data of the signing entity.

Note: As indicated in ISO/IEC 14888-1, depending on the application, there are different ways of forming the appendix and appending it to the message. The general requirement is that the verifier is able to relate the correct signature to the message. For successful verification, it is also essential that prior to the verification process, the verifier is able to associate the correct verification key with the signature.

6.2.8 Constructing the signed message

The signed message is obtained by the concatenation of message M and appendix,

$$M \,||\, ((R, S), \textit{text})$$

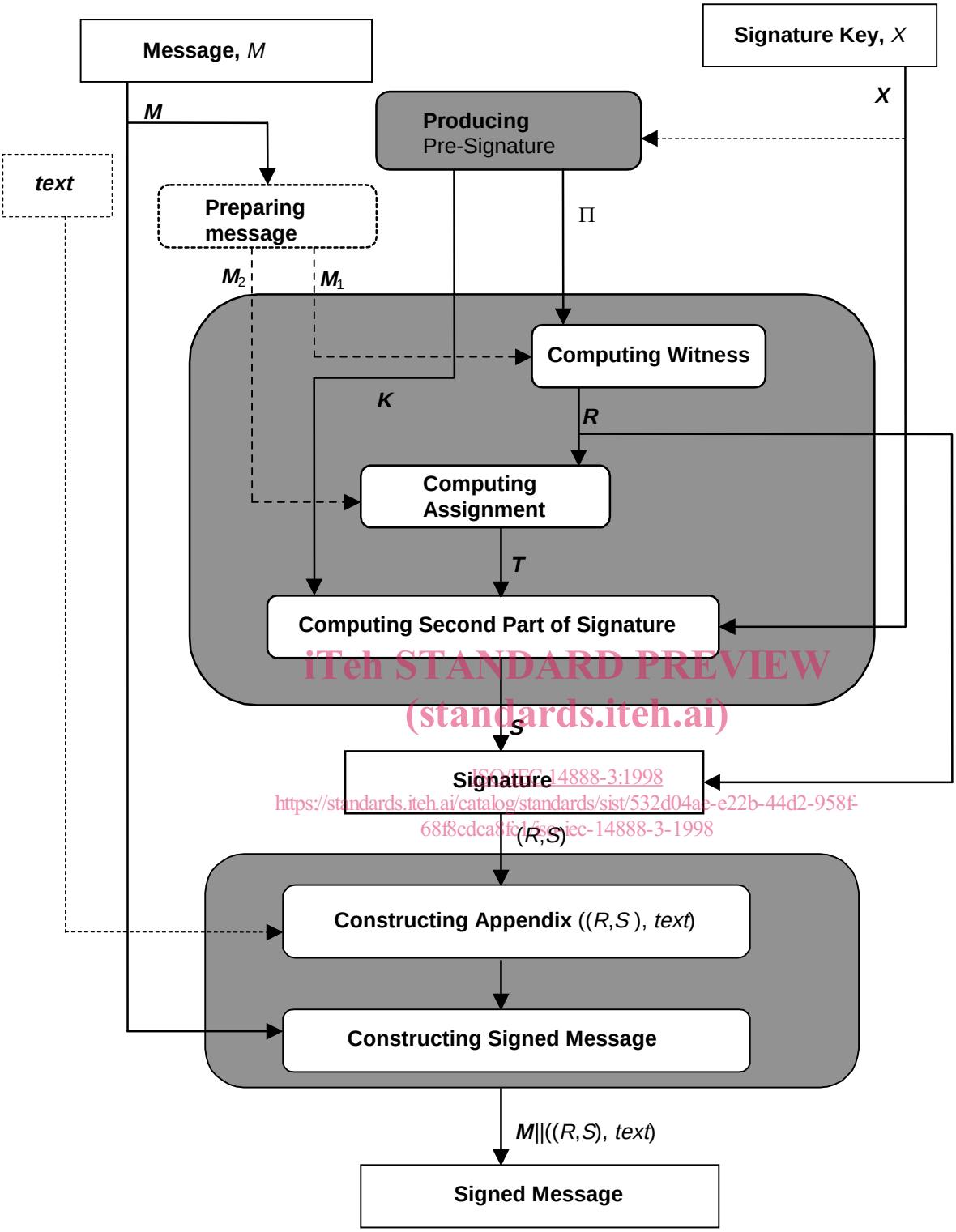

Figure 1 — Signature process with randomized witness

6.3 Verification process

The verification process consists of four stages (See Figure 2).

- Preparing message for verification
- Retrieving the witness
- Computing the verification function
  - retrieving the assignment
  - recomputing the pre-signature
  - recomputing the witness
- Verifying the witness.

In this process, the verifier makes use of the signer's verification key Y and the domain parameters: finite group ***E***, element G in ***E*** and its order Q.

6.3.1 Preparing message for verification

The verifier retrieves M from the signed message and divides the message into two parts M_1 and M_2.

6.3.2 Retrieving the witness

The verifier retrieves the signature (R, S) from the appendix, and divides it into witness R and the second part of the signature S.

6.3.3 Computing the verification function

6.3.3.1 Retrieving the assignment

This stage is identical to 6.2.5. The inputs to the assignment function consist of the witness R from 6.3.2 and M_2 from 6.3.1. The assignment $T = (T_1, T_2)$ is recomputed as the output from the assignment function.

6.3.3.2 Recomputing the pre-signature

The inputs to this stage are the set ***Z*** of domain parameters, the verification key Y, the assignment $T = (T_1, T_2)$ from 6.3.3.1 and the second part of the signature S from 6.3.2. The verifier assigns to the coefficients (A, B, C) the values (S, T_1, T_2) according to the order specified by the signature function, and computes the element $\overline{\Pi}$ in ***E*** as

$$\overline{\Pi} = Y^m \; G^n$$

where $m = -A^{-1} B \bmod Q$ and $n = -A^{-1} C \bmod Q$.

6.3.3.3 Recomputing the witness

The computations at this stage are the same as in 6.2.4. The verifier executes the witness function. The inputs are $\overline{\Pi}$ from 6.3.3.2 and M_1 from 6.3.1. The output is the recomputed witness, $\overline{R}$.

6.3.4 Verifying the witness

The signature is verified if the recomputed witness, $\overline{R}$ from 6.3.3.3 is equal to R from 6.3.2. Additional checks may be required (See A.1.2.4.6 for other example checks.)

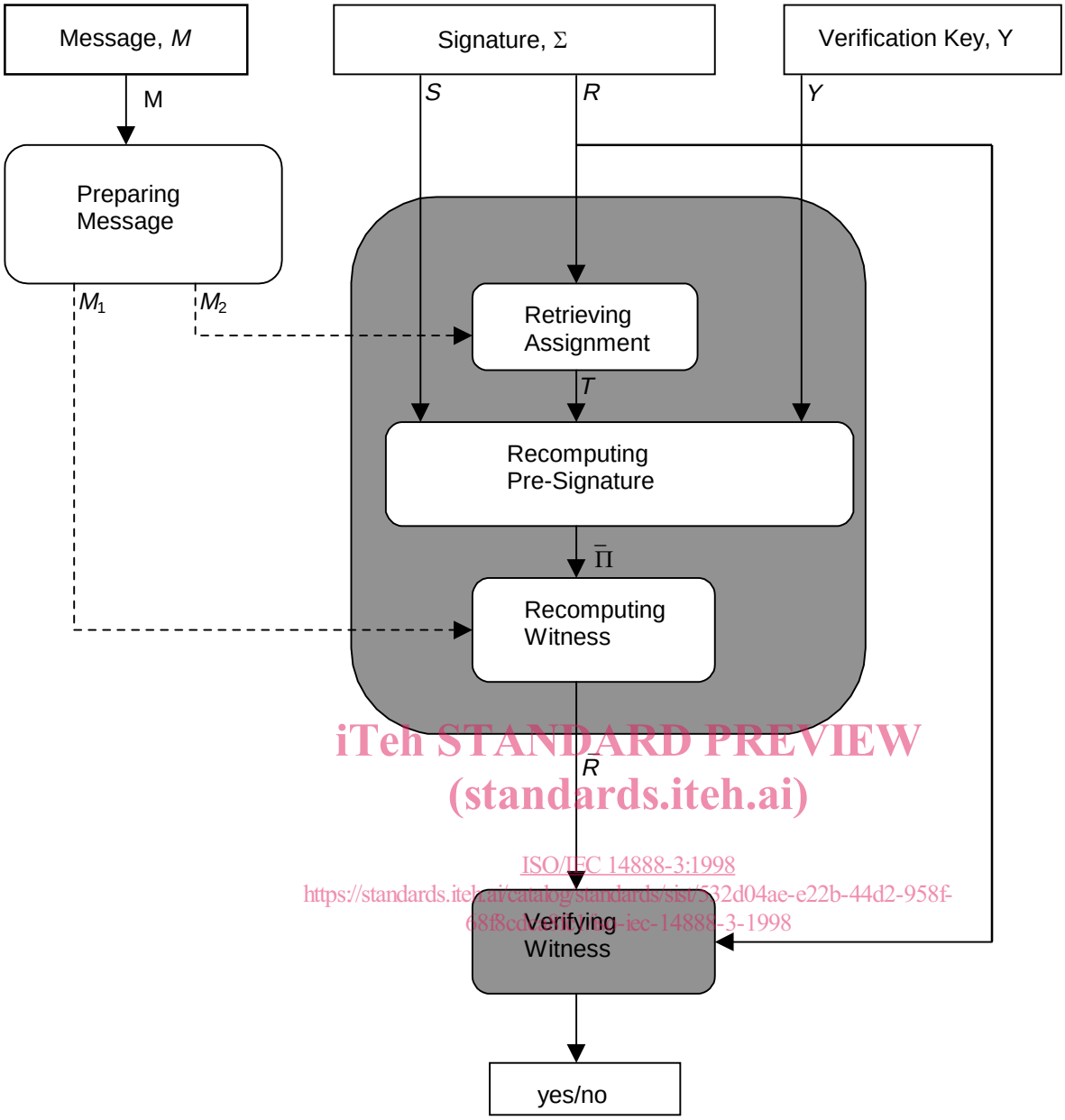

Figure 2 — Verification process with a randomized witness

7 Digital signature mechanisms based on factoring

Digital signature mechanisms based on factoring utilize a deterministic witness and produce a one-part signature, but can be randomized or deterministic (Reference ISO/IEC 14888-1, Figures 2 and 4). In either case, such a mechanism employs an integer N as a component of the verification key whose factorization is part of the signature key. It is assumed that it is computationally infeasible to factor N into its prime factors. Constraints should be imposed on the generation of the signature key to make the factorization sufficiently difficult.

7.1 Key generation process

7.1.1 Generation of domain parameters

For digital signature mechanisms based on factoring, the set $\boldsymbol{Z}$ of domain parameters optionally contains an integer v used as a system wide portion of the verification key, subject to the conditions specified in 7.1.2.

7.1.2 Generation of signature key and verification key

7.1.2.1 Generation of signature key

A signature key of a signing entity is a secretly generated collection $X = (\{P_1, P_2, \ldots, P_r\}, s)$, consisting of a set of randomly or pseudo-randomly chosen, but not necessarily distinct prime integers P_i, and an integer s. The minimum number of distinct primes to be used is two.

7.1.2.2 Generation of verification key

The verification key Y is a pair of integers (N, v) where N is the product, $\prod P_i$ of all primes P_i and v is an integer which satisfies a condition depending on the signature key.
If v is specified as a domain parameter, additional constraints might be imposed on the signature key so that v satisfies the appropriate condition.

7.2 Signature process

7.2.1 Producing the pre-signature (optional)

A randomized signature mechanism employs a pre-signature, which depends only on a randomizer and a signature key. The pre-signature is computed in two steps.

7.2.1.1 Producing the randomizer

The signing entity secretly generates a randomizer which is an integer K mod N, possibly subject to additional constraints. The output of this stage is K, which the signing entity keeps secret.

7.2.1.2 Computing the pre-signature

The pre-signature is a function of the randomizer and independent of the message. The input to this stage is the randomizer K and the signature key. The output of this stage is the pre-signature, denoted Π.

7.2.2 Preparing of message for signing

The message is used to construct data inputs M_1 and M_2. The second part, M_2, might be empty and the two inputs need not be distinct.

7.2.3 Computing the witness

The input to this stage is the data input M_1. The output is the hash token, H, determined by the data input M_1. Note that the hash token is interpreted as an integer mod N chosen so that $0 < H < N$.

7.2.4 Computing the signature

The inputs to this stage are the witness computed in 7.2.3, the signature key from 7.1.2.1 and optional data input M_2 (See ISO/IEC 14888-1, Figure 2). For a randomized mechanism, the randomizer K and the pre-signature Π are also valid inputs. The output is a one-part signature $\Sigma = S$.

7.2.5 Constructing the appendix

The appendix is constructed from the signature, Σ and an optional text field, *text*. The text field could include a certificate which cryptographically ties the public verification key to the identification data of the signing entity.

7.2.6 Constructing the signed message

The signed message is obtained by concatenating the message M with the appendix from 7.2.5,

$$M \,||\, (\Sigma, text).$$

7.3 Verification process

7.3.1 Preparing message for verification

The verifier retrieves M from the signed message and determines the two data input parts M_1 and M_2 as specified in 7.2.2.

7.3.2 Retrieving the witness

The verifier retrieves the value of the witness H as a function of the data input M_1 according to the witness function specified in 7.2.3.

7.3.3 Computing the verification function

Using the integer v obtained either from the domain parameter set $\boldsymbol{Z}$ or the verification key Y,

the verifier uses the verification function to obtain a recomputed witness, $\overline{H}$.

7.3.4 Verifying the witness

The signature is valid if the value of the retrieved witness H agrees with the value from the verification function of the recomputed witness, $\overline{H}$.

Annex A
(normative)

Examples of certificate-based digital signatures with appendix based on discrete logarithms

Examples of such signature mechanisms are the Digital Signature Algorithm (DSA) of the U.S. NIST, Pointcheval/Vaudenay, and elliptic curve signatures. These schemes are described below.

The groups used for the signature mechanisms include a multiplicative group $\mathbf{Z}^*_P$, where P is a prime (i.e., DSA and Pointcheval/Vaudenay) and an additive group formed by the points of an elliptic curve over a finite field (i.e., Elliptic Curve DSA).

A.1 Non-Elliptic curve based examples

A.1.0 Symbols and notation

P	prime integer
$\mathbf{Z}_P$	set of integers U with $0 \leq U < P$
$\mathbf{Z}^*_P$	set of integers U with $0 < U < P$

A.1.1 The U.S. Digital Signature Algorithm (DSA)

This example is taken from the U.S. National Institute of Standards and Technology (NIST) Federal Information Processing Standards Publication 186 (FIPS PUB 186), 19 May 1994. The general parameters defined in clause 6 shall have the following forms. The notation here has been changed slightly from FIPS PUB 186 to conform with notation used elsewhere in this part of ISO/IEC 14888.

The DSA is a signature mechanism with $\boldsymbol{E} = \boldsymbol{Z}^*_P$, P a prime, and Q a prime dividing P - 1. The message is split such that M_1 is empty and $M_2 = M$. The witness function is defined by the formula

$$R = \Pi \bmod Q$$

and the assignment function by the formula

$$(T_1, T_2) = (-R, -H)$$

where H = h(M) is the hash-token of message M, converted to an integer according to the conversion rule given in Annex C. The hash-function h is the Secure Hash Algorithm (SHA) as adopted in the U.S. NIST Secure Hash Standard (SHS), FIPS PUB 180-1, 17 April 1995. The Secure Hash Algorithm is also described in ISO/IEC DIS 10118-3. (Note that no control field with a hash-function identifier is required for DSA, thus the hash token is simply h(M). See ISO/IEC 14888-1).

The coefficients (A, B, C) of the DSA signature equation are set as follows

$$(A, B, C) = (S, T_1, T_2).$$

Thus the signature equation becomes

$$(SK - RX - H) \equiv 0 \pmod{Q}.$$

A.1.11 DSA Parameters

L	512 + 64l, for l an integer $0 \leq l < 8$
P	a prime, where $2^{L-1} < P < 2^L$
Q	a prime divisor of P-1, where $2^{159} < Q < 2^{160}$
F	an integer such that $1 < F < P$-1 and $F^{(P-1)/Q} \bmod P > 1$
G	$F^{(P-1)/Q} \bmod P$, an element of order Q in $\boldsymbol{E} = \boldsymbol{Z}^*_P$

The integers P, Q, and G can be public and can be common to a group of users.
To achieve FIPS compliance, parameters P and Q are generated as specified in FIPS PUB 186, Appendix 2 (Details can be found in Annex C of this part of ISO/IEC 14888).

Note 1: The size of the prime P in this normative example is as specified by the Digital Signature Algorithm (DSA). Note that the size of P is restricted to be at most 1024 bits. As of 19 May 1994, the size of P provides a sufficient security margin. It is acknowledged that future advances in number theoretic algorithms may possibly render the size of P of 1024 bits as insufficient.

Note 2: It is recommended that all users check the proper generation of the DSA public parameters.

Note 3: It is recognized that DSA possesses an unfavourable property in which an attack can be mounted where collisions on the underlying hash function can be found with a complexity of 2^{74} as compared to 2^{80} in the most secure case. This attack though is easily detectable. For users who may still wish to avoid this property, it can be prevented by using the mechanism of A.1.2.

A.1.12 DSA generation of signature key and verification key

The signature key of a signing entity is a secretly generated random or pseudo-random integer X such that $0 < X < Q$. The corresponding public verification key Y is

$$Y = G^X .$$

A user's secret signature key X and public verification key Y are normally fixed for a period of time. The signature key X must be kept secret.

A.1.1.3 DSA signature process

A.1.1.3.1 Producing the randomizer

The signing entity computes a random or pseudo-random integer K such that $0<K<Q$. Parameter K must be generated for each signature and must be kept secret.

A.1.1.3.2 Producing the pre-signature

The input to this stage is the randomizer K and the signing entity computes

$$\Pi = G^K \bmod P$$

A.1.1.3.3 Preparing the message for signing

The message is split such that M_1 is empty and M_2 is the message, $M_2 = M$.

A.1.1.3.4 Computing the witness

The signing entity computes $R = \Pi \bmod Q$ where the witness is simply a function of the pre-signature. Thus,

$$R = (G^K \bmod P) \bmod Q$$

A.1.1.3.5 Computing the assignment

The signing entity computes the assignment $(T_1,T_2) = (-R,-H)$ where $H = h(M)$ is the hash-token of message M and $M = M_2$.

A.1.1.3.6 Computing the second part of the signature

The signature is (R, S). Thus,

$$R = (G^K \bmod P) \bmod Q$$
$$S = (K^{-1}(h(M) + XR)) \bmod Q$$

The value of $h(M)$ is a 160-bit string output of the Secure Hash Algorithm. For use in computing S, this string must be converted to an integer. The conversion rule is given in Annex C.

As an option, one may wish to check if $R = 0$ or $S = 0$. If either $R = 0$ or $S = 0$, a new value of K should be generated and the signature should be recalculated. (It is extremely unlikely that $R = 0$ or $S = 0$ if signatures are generated properly).

A.1.1.3.7 Constructing the appendix

The appendix will be the concatenation of (R, S) and an optional text field, *text*, $(R, S)||text$.

A.1.1.3.8 Constructing the signed message

A signed message is the concatenation of a message, M, and the appendix.

$$M||(R, S)||text$$

A.1.1.4 DSA verification process

Prior to verifying the signature of a signed message, it is necessary that the verifier has trusted copies of P, Q and G.

The verifier also acquires the necessary data items for the verification process. For example, the verification key (see ISO/IEC 14888-1, clause 9 for additional required data items).

A.1.1.4.1 Preparing the message for verification

The verifier retrieves $M = M_2$ from the signed message. M_1 is empty.

A.1.1.4.2 Retrieving the witness

The verifier retrieves the witness R and the second part of the signature S from the appendix.

A.1.1.4.3 Retrieving the assignment

This stage is identical to A.1.1.3.5. The inputs to the assignment function consist of the witness R from A.1.1.4.2 and M_2 from A.1.1.4.1. The assignment $T = (T_1,T_2)$ is recomputed as output from the assignment function, A.1.1.3.5.

A.1.1.4.4 Recomputing the pre-signature

The inputs to this stage are domain parameters, verification key Y, assignment $T = (T_1,T_2)$ from A.1.1.4.3 and second part of the signature S from A.1.1.4.2. The verifier assigns the coefficients (A, B, C) the values (S,T_1,T_2) as determined by the signature function, and obtains a recomputed value $\overline{\Pi}$ of the pre-signature using the formula

$$\overline{\Pi} = Y^{-A^{-1}B \bmod Q} G^{-A^{-1}C \bmod Q} \bmod P \text{ in } \mathbf{E}.$$

A.1.14.5 Recomputing the witness

The computations at this stage are the same as in A.1.1.3.4. The verifier executes the witness function. The input is $\overline{\Pi}$ from A.1.1.4.4. Note that