INTERNATIONAL STANDARD

ISO/IEC 14888-2

First edition
1999-12-15

Information technology — Security techniques — Digital signatures with appendix —

Part 2: Identity-based mechanisms

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

Partie 2: Mécanismes basés sur des identités

Reference number
ISO/IEC 14888-2:1999(E)

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. 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 14888-2 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 of this part of ISO/IEC 14888 are for information only.

Information technology — Security techniques — Digital signatures with appendix —

Part 2:
Identity-based mechanisms

1 Scope

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

This part of ISO/IEC 14888 specifies the general structure and the fundamental procedures which constitute the signature and verification processes of an identity-based digital signature mechanism with appendix for messages of arbitrary length.

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 9796: 1991, *Information technology - Security techniques - Digital signature scheme giving message recovery*.

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

3 General

The verification of a digital signature requires the signing entity's verification key. It is therefore essential for a verifier to be able to associate the correct verification key with the signing entity, or more precisely, with (parts of) the signing entity's identification data. If this association is somehow inherent in the verification key itself, the digital signature scheme is said to be "identity-based."

The key generation process of the identity-based mechanism defined in this part of ISO/IEC 14888 involves a trusted third party. The trusted third party has a secret parameter, the key generation exponent, which it uses to derive the signature keys of other entities. The secrecy of the signature keys depends unconditionally on the secrecy of the key generation exponent.

In the verification of an identity-based signature, two parameters are needed. The first one, the domain verification exponent, is common to all entities, while the second one, the signing entity's verification key, is specific to each entity. In the identity-based mechanism defined in this part of ISO/IEC 14888, an entity's verification key is derived directly from the entity's identification data, using a public function.

The identity-based signature mechanism with appendix is an example of a randomized mechanism, as described in ISO/IEC 14888-1. The descriptions of the signature and verification processes follow the general procedures specified in Clause 10 of ISO/IEC 14888-1. In particular, this part makes use of the general requirements, definitions and symbols given in ISO/IEC 14888-1.

The identity-based digital signature mechanism with appendix is defined by the specification of the following processes:

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

4 Definitions

For the purposes of this part of ISO/IEC 14888, the following definitions apply.

4.1
domain modulus
a domain parameter, which is a positive integer resulting from the product of two distinct primes which are known only to the trusted third party

4.2
domain verification exponent
a domain parameter which is a positive integer

4.3
key generation exponent
a positive integer known only to the trusted third party

4.4
public key derivation function
a domain parameter, whose function is to map strings of bits into positive integers

NOTE 1 This function is used to transform an entity's identification data into the entity's verification key, and satisfies the following two properties.

- It is computationally infeasible to find any two distinct inputs which map to the same output.
- Either the probability that a randomly chosen value Y is in the range of the function is negligibly small, or for a given output it is computationally infeasible to find for a given output an input which maps to this output.

NOTE 2 Negligibility and computational infeasibility depend on the specific security requirements and environment.

4.5
trusted third party
A security authority, or its agent, trusted by other entities with respect to security related activities

5 Notation

5.1 Symbols

In addition to the symbols defined in ISO/IEC 14888-1, the following symbols are used.

D	Key generation exponent
I	Identification data
N	Domain modulus
P, Q	Prime numbers
V	Domain verification exponent
y	Public key derivation function
lcm (A, B)	The least positive integer
C	such that C mod A = 0 and
C	≡ 0 (mod B)

6 Key production process

The key production process of this identity-based signature mechanism is an application of the signature scheme specified in Annex A of ISO/IEC 9796-1. It consists of the following three procedures:

- generating domain parameters; and
- generating signature key.

An entity shall be chosen to act as a trusted third party. Using its own secrets, the trusted third party generates the private signature key for each entity as a function of that entity's identification data.

In some instances the validation of domain parameters and keys, may be required. However, it is outside the scope of this part of ISO/IEC 14888.

6.1 Generating domain parameters

This procedure is executed once when the domain is set up.

The trusted third party produces the domain verification exponent V and the domain modulus N. The domain verification exponent shall be chosen to be an odd integer. The domain modulus shall be the product of two large prime integers P and Q such that $P - 1$ and $Q - 1$ are coprime to V. Furthermore, the trusted third party determines the key generation exponent, which is the least positive integer D such that $DV - 1$ is a multiple of lcm ($P - 1$, $Q - 1$).

Specifically, D shall be such that:

$$U^{DV} \bmod N = U$$

for all integers U, $0 < U < N$.

The public domain parameters are N and V. The trusted third party keeps D for its own use. It shall be computationally infeasible for other entities to compute D from N and V.

NOTE — The values of N and V shall be chosen large enough to satisfy the specific domain security requirements. The length of N varies typically between 1024 bits and 2048 bits while the length of V is recommended to be at least 80 bits.

Let T, T' and L be integers such that $T - T' = LV$. Let X be a signature key and Y be the corresponding verification key as defined in 6.2. Then $X^{T'} \equiv X^T \cdot Y^L \bmod N$.

Given a signature (R, S) corresponding to assignment T, one can then produce the signature (R, S') corresponding to assignment T' by computing $S' = S \cdot Y^L \bmod N$ (See 7.4). Therefore, V must be chosen sufficiently large so that given T, the probability that a randomly chosen T' satisfies $T - T' \equiv 0 \bmod V$ is sufficiently small.

The set of the domain parameters of an identity-based signature mechanism with appendix also includes a public key derivation function y, which is used to transform a signing entity's identification data into a positive integer less than N.

6.2 Producing the verification key and generating the signature key

Each user entity has unique identification data. To generate a private signature key for an entity with identification data I, the trusted third party first

computes the verification key Y from the public key derivation function y and the identification data I.

$$Y = y(I).$$

NOTE — The probability that Y is equal to zero or an integer multiple of P or Q is negligible, provided that P and Q are sufficiently large.

The trusted authority computes the private signature key X as

$$X \equiv Y^{-D} \bmod N.$$

As a result of the key generation process, the entity with identification data I has a signature key X which satisfies the equation,

$$X^{V} \cdot y(I) \bmod N \equiv 1.$$

In an identity-based signature mechanism, the verifier obtains the knowledge of the signing entity's verification key from the signing entity's identification data by computing $Y = y(I)$. Once the verification has been computed, it can be cached for further use.

7 Signature process

In this clause, the signature process of an identity-based signature mechanism is described. The mechanism is randomized and follows the general model for randomized signature mechanisms described in ISO/IEC 14888-1. The signature process consists of the following procedures:

- producing pre-signature;
- preparing message;
- computing witness; and
- computing signature.

In this process the signer makes use of its signature key X and the public domain parameters N and V.

The result of this process is the signature Σ which has two parts, R and S. The signing entity optionally forms a text field, containing the entity's identification data. The signature and the optional text field form the appendix, which is appended to the message by the signer.

7.1 Producing pre-signature

The pre-signature procedure of an identity-based signature mechanism consists of two steps:

- producing randomizer K; and
- computing pre-signature Π.

7.1.1 Producing randomizer

The signing entity generates a randomizer, which is an integer, K with $0 < K < N$. Depending on the mechanism, there may be additional constraints on this generation procedure. The output of this step is K, which the signing entity keeps secret.

7.1.2 Computing pre-signature

The input to this step is the randomizer K. The output is the pre-signature Π computed as:

$$\Pi = K^{V} \bmod N.$$

7.2 Preparing message

Two data fields, M_1 and M_2 are derived from the message M as described in 8.2 of ISO/IEC 14888-1.

The process of preparing the message shall satisfy one of the following two conditions:

- the entire message shall be re-constructible given M_1 and M_2; or
- it shall be computationally infeasible to find two distinct messages M and M' such that the derived pairs (M_1, M_2) and (M'_1, M'_2) are equal.

Typically in the first case, $M_1 = M$ (when M_2 is empty), or $M_2 = M$ (when M_1 is empty), or $M_1 = M_2 = M$. In the second case, either M_1 or M_2 or both are hash tokens of M.

7.3 Computing witness

The deterministic witness is computed as a hash-token H of M_1 using a collision-resistant hash-function (see Figure 1).

The randomized witness R is defined as the concatenation of an optional field, which can be used to identify the hash-function and padding method and is computed using a collision-resistant hash-function. R depends on the pre-signature Π and optionally, on M_1, if $M1$ is not empty (see figure 2).

NOTE — The hash-function identifier shall be included in the hash-token unless the hash-function is uniquely determined by the signature mechanism or by the domain parameters.

7.4 Computing signature

The computation of a signature in an identity-based signature mechanism consists of the following steps:

- computing the first part of the signature;
- computing assignment; and
- computing the second part of the signature.

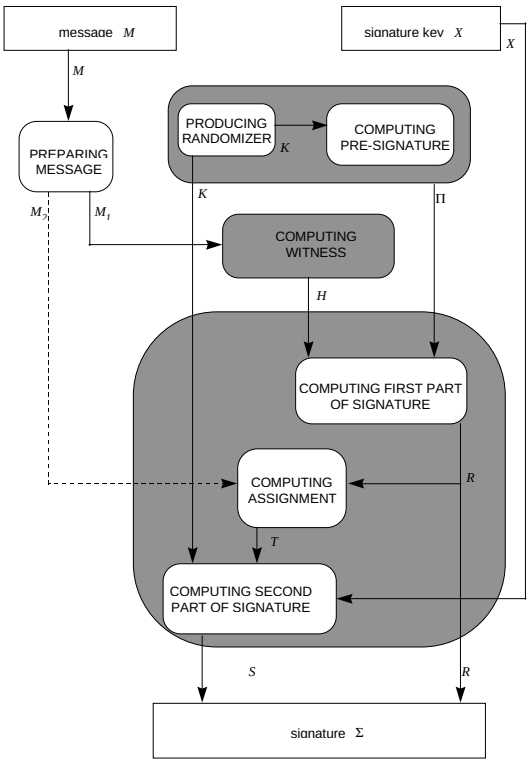

Figure 1 — Signature process with a deterministic witness

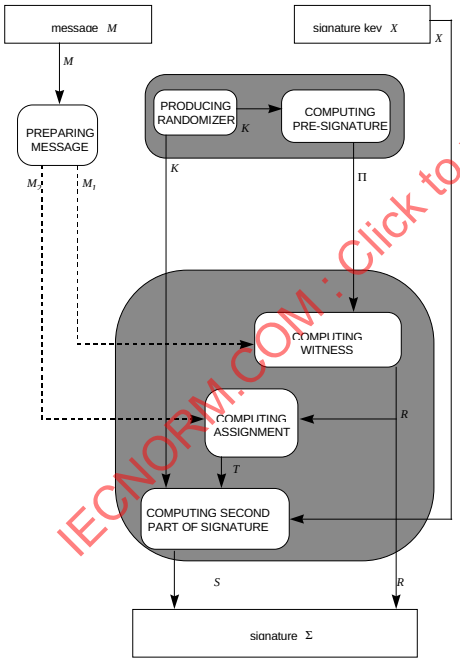

Figure 2 — Signature process with a randomized witness

7.4.1 Computing the first part of the signature

If the witness is deterministic, the first part R of the signature is computed as a function of H and Π. This function is invertible in the following way.

- Given Π and R, the hash-token H can be recovered (see Figure 1).

If the witness is randomized, it is the first part of the signature R and no further computation is needed (see Figure 2).

NOTE — The domain parameters must include an agreed upon method for converting a string of bits to a positive integer, in order that this step can be carried out.

7.4.2 Computing assignment

The assignment T is a positive integer and computed as a function of the first part of the signature. The assignment also depends on the second part M_2 of the message, if M_2 is not empty.

It is required that the method of computing the pair (R, T) satisfies the following condition.

It is computationally infeasible to find any two pairs (M, Π) and (M', Π') with the same resulting pair (R, T).

7.4.3 Computing the second part of the signature

The signature function of the identity-based signature mechanism has the following form,

$$S = K \cdot X^T \bmod N,$$

where K is the randomizer computed in 7.1.1, T is the assignment computed at 7.4.2, X is the signature key, and N is the domain modulus. The output of the signature function is the second part S of the signature.

8 Verification process

The verification process consists of the following procedures:

- preparing message;
- retrieving witness;
- computing verification function; and
- verifying witness.

At the beginning of the verification process the verifier must have the values of the following data items available:

- domain parameters N and V;
- the signer's verification key Y;
- message M;
- signature $\Sigma = (R, S)$; and
- optional text (from appendix).

A successful verification of a signature implies that the signature was created using a signature key X which corresponds to the verification key Y.

8.1 Preparing message

This procedure is identical to 7.2.

8.2 Retrieving witness

If the witness is deterministic, then this procedure is described in 7.3. The verification process is depicted in Figure 3.

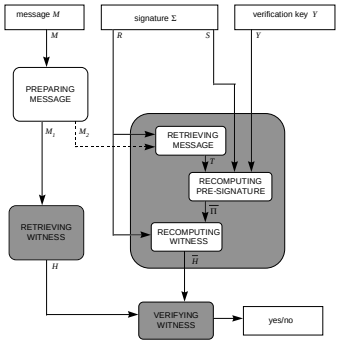

Figure 3 — Verification process with a deterministic witness

If the witness is randomized, the verification process is as depicted in Figure 4. The retrieved witness is the first part R of the signature.

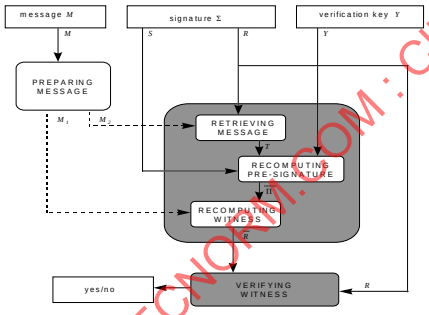

Figure 4 — Verification process with a randomized witness

8.3 Computing verification function

The computation of the verification function of an identity-based signature mechanism consists of the following steps:

- retrieving assignment;
- recomputing pre-signature; and
- recomputing witness.

8.3.1 Retrieving assignment

This step is identical to 7.4.2. The verifier computes the assignment T as a function of the value R retrieved in 8.2 and the part M_2 of the message if M_2 is not empty. The assignment is a positive integer.

8.3.2 Recomputing pre-signature

The verifier produces a recomputed value $\overline{\Pi}$ of the pre-signature by using the formula,

$$\overline{\Pi} = Y^T \cdot S^V \bmod N$$

where Y is the verification key, N is the domain modulus, V is the domain verification exponent, T is the assignment retrieved at 8.3.1 and S is the second part of the signature.

8.3.3 Recomputing witness

If the witness is deterministic, it is the hash-token H of M_1. The verifier produces a recomputed value $\overline{H}$ by recovering it from using R and $\overline{\Pi}$.

If the witness is randomized, the computation is identical to 7.3, where the inputs are the recomputed value $\overline{\Pi}$ of Π and M_1 of M. The output is the recomputed witness $\overline{R}$.

8.4 Verifying witness

At this step the retrieved value of the witness from 8.2 and the recomputed value of the witness from 8.3.3 are compared. The verification is successful if these two witness values are equal.

9 Guillou-Quisquater signature mechanism

The key production process, signature process and verification process of the identity-based digital signature scheme of Guillou and Quisquater are described in Clauses 6 to 8 of this part of ISO/IEC 14888 and depicted in Figures 2 and 4.

The following procedures and functions remain to be specified in more detail.

- public key derivation function;
- preparing message;
- computing witness;
- computing the first part of the signature; and
- computing assignment.

9.1 Public key derivation function

The public key derivation function is the redundancy generating function described in ISO/IEC 9796-1. The signing entity's identification data I, or its hash-token, is taken as the input message to the process described in 5.1 to 5.4 of ISO/IEC 9796-1. The public key $Y = y(I)$ of the entity is set to be the output of this process, which is called the intermediate integer in ISO/IEC 9796-1.

9.2 Preparing message

In the Guillou-Quisquater mechanism $M_1 = M$ and M_2 is empty.

9.3 Computing witness

The witness R in the Guillou-Quisquater mechanism is computed as a hash-token of the data $\Pi \| M$, where Π is the pre-signature.

$$R = H(\Pi \| M).$$

9.4 Computing the first part of the signature

In the Guillou-Quisquater mechanism, the witness R forms the first part of the signature.

9.5 Computing assignment

The assignment T is equal to R represented as a positive integer.

NOTE — The domain parameters must include an agreed upon method for converting a string of bits to a positive integer in order that this step can be carried out.

10 Identity-based signatures with short assignment

In this clause a variant of the Guillou-Quisquater scheme is specified, which is suitable for implementation in devices with slow processors or slow input/output interface. Additional hashing is used to reduce the length of the intermediate parameters. Up to three different hash-functions can be used in the computation of the witness and the assignment. They need not be collision-resistant, but they have to be chosen in such a way that it is infeasible to find two distinct messages with the same pair (R, T).

This mechanism differs from the Guillou-Quisquater scheme in the following procedures:

- preparing message;
- computing witness;
- computing the first part of the signature;
- computing the first part of the signature; and
- computing assignment.

10.1 Preparing message

In this variant, the two parts M_1 and M_2 of the message input are set to equal the hash-token H of the message M.

10.2 Computing witness

The witness R is computed by forming the hash-token H_2 of the data $H_1 \| H$, where H is computed in 10.1 and is the hash-token of the pre-signature Π.

NOTE — The pre-computed hash-token H_1 can be stored more efficiently.

10.3 Computing assignment

The assignment T is obtained by computing the hash-token H_3 of the data $H \| R$.

11 Identity-based signatures giving recovery of the hash-code of message

This variant of the Guillou-Quisquater scheme has the advantage that the hash-function can be computed in parallel to the rest of the verification steps. The signature process of this identity-based digital signature mechanism is depicted in Figure 1 and the verification process in Figure 3. The witness is deterministic. This mechanism differs from the Guillou-Quisquater scheme in the following procedures:

- computing witness; and
- computing the first part of the signature.

Also, it is assumed that the randomizer K produced by the signer satisfies $K \neq 0 \bmod P$ and $K \neq 0 \bmod Q$, since otherwise the factorization of N is found. Hence it can also be assumed that the pre-signature Π is not a multiple of either P or Q.

11.1 Computing witness

The witness is computed as the hash-token H of the message M using a collision-resistant hash-function.

11.2 Computing the first part of the signature

The first part R of the signature is computed as;

$$R = \Pi \cdot H \bmod N.$$

Given a recomputed value $\overline{\Pi}$ of the pre-signature, the verifier obtains a recomputed value $\overline{H}$ of the witness by computing:

$$\overline{H} = \overline{\Pi}^{-1} R \bmod N.$$

Annex A
(informative)

Numerical examples

The numbers are given in hexadecimal notation.

A.1 Numerical example of the key production process

A.1.1 Generating domain parameters

The domain modulus N is the product of two distinct primes P and Q. This example uses 512-bit prime numbers as P and Q, so N is a 1024 bit number.

P = FFFFFFFF EA2DE66E D3B1B7E9 61B75DFC D9FAE2FF
A07A2345 9B7956FB 1B9B16D7 E1B6D59B BDF45B85
3CBF08EA 3BC7A1BD 541CB3A8 80E02E43 87CA7DEF
50948E87

Q = FFFFFFFF E275B7F4 98A3811D E906ACF7 BFEB5CD6
A445AF09 D7906DE1 97CC2CCD 87614718 8C7C084F
CE9231CA B7CFA113 13C3DDCF F1B70A54 84494467
8FCEF193

$N = PQ$ =
FFFFFFFF CCA39E63 6ED9CF52 950C23A0 38AE0291
012B984A 964FFBBD 99E9DACB 91400431 0C5DD264
B1873126 44A725C5 D5BC73F4 97CFD100 89FD1342
656026BE 3FB583FE B134FF43 6957A1E1 D975B5BE
DF1A9570 4C81A337 706E5F9F 9388A7AC 5ABFD5CF
0356D91A 9861C69F E50509C2 323E5270 F2015FBD
C08AA2C0 391CEE85

The domain verification exponent is an odd integer coprime to P-1 and Q-1. In this example, V is 2^{79}+1, so the length of V is 80 bits.

V = 8000 00000000 00000001

The key generation exponent D is the positive integer such that DV - 1 is a multiple of lcm(P-1,Q-1).

lcm(P-1, Q-1) =
7FFFFFFF E651CF31 B76CE7A9 4A8611D0 1C570148
8095CC25 4B27FDDE CCF4ED65 C8A00218 862EE932
58C39893 225392E2 EADE39FA 4BE7E880 44FE89A1
32B0135E 1FDAC1FF 7248B06F FE81346D 475BD565
229A2ACD 03E0E874 3EB24D61 7010B203 78D3DC8D
5C733AA2 C68845F5 78B6E378 E52EE07C 3FB51392
DA3B7034 AC5CB736

D = 1BC6C0ED 36435CBF A89C7A35 50CE3D54 C6ABC9F5
EE5E75C9 E458AADA 6178CB20 C7339C4E F30413A6
586DA8B6 45A72BDF 291C9218 F0CA83EF A4234FAD
8394B2BF 8F4A0EF9 61E098FC 2CC5AFAA 46CCC821
0427D3EE 3461AF0B 46895311 E1DAD21F 35217CBC
4FD1A5B9 62E01B8B 967F97E2 41ECF56E DBF85278
EC058601 17D9A7B7

A.1.2 Producing verification key and signature key

The verification key Y is the output of the public key derivation function y with the identification data I as an input, and neither a multiple of P nor Q. This example uses a 1024-bit number Y less than N.

$Y =$

```
C50ECCC9  64443B0A  1C974F40  1C94E500  FA8214FC
9B1B5EC5  2AA1201A  001EA099  FE90D01D  F32C6B43
323F0812  42ABE843  09F926BB  9338A841  5DEF2EF6
E709E3BD  515B5D86  C3ED4B7F  C15FA876  26E8E9C7
0E557D5B  A8E96D7C  B55FBF41  37F601FF  47B7CCCB
6BED4407  6F8E9805  42E37105  522E7184  42A717DF
E89A6B62  7B6E60B7
```

The Trusted authority computes the private signature key X as:

$X = Y^{-D} \bmod N =$

```
A763FA4   3895CFDD  D80627A6  A8271250  97C184E5
10F0075C  48FCB0E7  F2885275  AAA32829  C08CF352
0F42F6FD  C296DCE1  F50FBDED  D5C33C7C  63298C4E
26C2CDEE  11D927BA  C6EC4A6A  C022C063  1F30E880
07452397  7F3ACA8C  422E2461  3B7F3BB0  E61D04B8
0670A128  0ED7C8C1  A72D4B1C  C566381B  0605F83B
70FD7158  0B7A6EEC
```

A.2 Numerical example of a Guillou-Quisquater signature mechanism, described in clause 9

In this example, the hash-function in use is fixed and known to all entities of a domain and hence the hash identifiers are not required. The hash-token is produced by using SHA-1.

The domain parameters, the private signature key and the verification key are the same as A.1.

A.2.1 Signature process

A.2.1.1 Producing pre-signature

The signing entity generates a randomizer, which is a random or pseudo-random integer K with $0 < K < N$. This example uses a 1024-bit randomizer K.

$K =$

```
B2045A19  83150F5B  B04CB524  2B566A37  6779F416
5C7F1673  029C1BFC  05A84C60  E401897A  CEAD9DE5
C7D8108B  95943332  FF6B20D3  004CCD40  36BDBB7E
10DA755E  B03720F0  5A0CDC53  66EB4374  BE091A80
6D339190  D2ADE1CD  9E11EF4E  A5FF6969  3D0EB942
BFCC333D  F5FDD599  1D3B78A7  4868B0C6  381AEA61
C24F3E05  2D8D9FFA
```

The pre-signature Π is computed as:

$\Pi = K^{V} \bmod N =$

```
15B35BB7  0DDD7ED8  0FAD7EDE  A80F828E  46B3F86D
4EFB7E84  58562B6D  6F1885D0  A02FD892  8838C128
B53EE703  FAC96534  6C18A714  17D12FD7  211C3956
B6AD1A15  F4399EB0  CC065E8F  CC0039F3  E13A8EFE
5FB384CC  D0190FA9  DC995DEC  BB07947F  72124D7B
9044433C  7A416B62  99297387  0174FB6A  94A8FA8B
1CBD7E0A  2780DFD8
```

A.2.1.2 Preparing message

In this example, the message is as following text sentence.

M: This is a test message!

A.2.1.3 Computing witness

In this example, the witness forms the first part of the signature.

The first part R of the signature:

R = hash-token of $\Pi||M$ =

```
42AF9098    F1596D17    EAC6EA31    36E7D9F3    30F8E9FB
```

A.2.1.4 Computing assignment

The assignment T is equal to R represented as a positive integer.

$T = R$

A.2.1.5 Computing the second part of the signature

The second part S of the signature is computed as:

$S = K \cdot X^T \bmod N$ =

```
B1FAF642    59F99E7A    B32029A6    1FDEE247    F068853F
80C6FF87    9FB07983    A61C047B    34CEE1C8    F69FF97B
020F0B6C    F37E4A85    05EAEF00    01E825E5    8B8F3438
0C332BF5    B3D47E5F    C654747F    E1289D61    061F124B
C19EBB38    D466970C    39CEC404    703FB359    A0019692
F968F760    72F2D6F5    0CD75C73    1D9EEE7F    EFFB6F98
E80BD095    5D00C051
```

The signature Σ is mapped to (R, S).

A.2.2 The verification process

The verifier has the values of the following data items available:

$N, V, Y, M, \Sigma = (R, S)$.

A.2.2.1 Retrieving assignment

The assignment T is equal to the first part R of the signature.

$T = R$

A.2.2.2 Recomputing pre-signature

The verifier produces a recomputed value $\overline{\Pi} = Y^T \cdot S^V \bmod N$ of the pre-signature.

$\overline{\Pi} = Y^T \cdot S^V \bmod N$

NOTE: If ($Y^T \bmod N$) and ($S^V \bmod N$) are calculated separately, they are given as:

Y^T mod N =

```
3C18CAA0  339ED6CA  6C80AF2C  201481C1  74054C22
A8314537  0B8C6DF1  92766BC4  C3FD8C14  76EDD630
E02E33E8  4F557C86  A51FE7B5  9769CF40  98E34D29
111E9BF0  82825727  85B047F3  82F51DB4  51558F87
FE5FAA6F  C1C45803  77AC051D  85A094BD  65472145
4618DECB  EC2B58FF  891DC06F  D202E815  1A39ACA5
8D5C60D9  419695E7
```

S^V mod N =

```
A8E6DDCF  74D3C94D  23F5FDE0  820431B4  FA51F3D9
150D5CDE  DB0692A0  735FD729  BCAC0825  7CA19BA1
565F03BA  7849B538  B9BF1797  5473FC28  299F0F3C
04C3EE35  E31BFA2C  A17F1781  0488F988  05439C3E
7B8BE70C  F027401D  786DAC48  AE2507F5  30EC01C7
1090504D  9BD82673  CD472405  8EF34697  21F90D3F
1E2DD798  50AC0090
```

A.2.2.3 Recomputing witness and verifying witness

The witness is recomputed as the hash-token $\overline{\Pi} = Y^T \cdot S^V \bmod N \| M = R$.

The recomputed witness $\overline{\Pi} = Y^T \cdot S^V \bmod N$ is the same as the retrieved witness R.

A.3 Numerical example of identity-based signatures with short assignment, described in clause 10

In this example, the hash-functions in use are fixed and known to all entities of a domain and hence the hash identifiers are not required. The hash-token H, H_1, and H_2 are produced by using SHA-1. The other hash-token H_3 is computed by the following function h with the two 160 bit inputs that produces a 80 bit output.

$h(u||v) = \{(u_1 \oplus u_2)+(v_1 \oplus v_2)\} \bmod 2^{80}$, for $u = u_1 \cdot 2^{80}+u_2$ and $v = v_1 \cdot 2^{80}+v_2$ where $\oplus$ is a bit-by-bit exclusive-or operation.

The domain parameters, the private signature key and the verification key are the same as A.1.

A.3.1 Signature process

A.3.1.1 Producing pre-signature

The signing entity generates a randomizer, which is a random or pseudo-random integer K with $0 < K < N$. This example uses a 1024-bit randomizer K and the corresponding pre-signature Π is given in A.2.1.1.

H_1 is pre-computed and saved in a storage to compute the witness more efficiently.

H_1 = hash-token of the pre-signature Π =

```
7B2B1483  46B8D0B0  F42C762B  FE83D691  C9BC3841
```

A.3.1.2 Preparing message

In this example, the message is same as one in A.2.1.2.

H = hash-token of the message M =

```
A9D66D4B  652597FB  32DD1092  E7C9CDE1  8F0C7FBC
```

A.3.1.3 Computing witness and the first part of the signature

In this example, the witness forms the first part R of the signature.

$R = H_2$ = hash-token of $H_1||H$ =

```
D6149DF3  DA7E60EA  817CAADE  D2F9F785  8800903F
```

A.3.1.4 Computing assignment

The assignment T is obtained by computing the hash-token H_3 of the data $R \| H$.

$T = h(H||R) =$ 360E D98CD6C0 01E15EA4

A.3.1.5 Computing the second part of the signature

The second part S of the signature is computed as:

$S = K \cdot X^T \bmod N =$

FA17252E	94A3D992	C5CE7953	D0939F9E	45A4AE8A
A9F2B89F	1345528F	658959C9	B5B16E92	9131A3A0
BFA8490C	E4652452	90ECCFDF	73E67220	7D34AFA0
F0B1C381	EDE82B2C	2B2D520E	4E91EB67	98922DF5
04B1883E	D7CD9AC3	1E3E4E99	FF2E57F9	ACE6F5E6
745A029E	1821081F	02DB73ED	6640A148	EF281E9B
63E3B7E1	DA0E9AA9			

The signature Σ is mapped to (R, S).

A.3.2 The verification process

The verifier has the values of the following data terms available:

$N, V, Y, M, \Sigma = (R, S)$.

A.3.2.1 Preparing message

H = hash-token of M

A.3.2.2 Retrieving assignment

The assignment T is obtained by computing the hash-token H_3 of the data $R \| H$.

$T = h(R||H)$

A.3.2.3 Recomputing pre-signature

The verifier produces a recomputed value $\overline{\Pi} = Y^T \cdot S^V \bmod N$ of the pre-signature.

If ($Y^T \bmod N$) and ($S^V \bmod N$) are calculated separately, they are given as:

$Y^T \bmod N =$

390AB7EB	69A80EE0	D382FFA6	6D6557B3	D17D68A4
CE2D67F1	FCF76F27	D660941A	90941033	D3269612
DC989EA4	78257463	8C98ADEA	F05DAC71	87A000BE
C7E20991	9B0598D1	49B22923	6C45DF67	358E4DEF
900DC37C	8812CE7A	D827A550	3703BC3C	DE843E4E
B12C7C16	DBB7AD5B	97F4F0D9	D4918CA5	32060A48
E58A9445	437318E9			

$S^V \bmod N =$

3A65050A	97021E9A	8A6E825A	26EE068D	EF06D046
0617D9FE	3C577159	06C15DD0	2EBF8707	CC681798
373B4E16	9A12A839	04B7CBAF	D578BC4E	7D0FD50D
0C7E549C	CECC18BA	A198682F	CB46844A	E5C904E4
A9003E35	A0035D96	B170F5A7	E1C5B285	4F66C7BD
8132EB24	A9F3EDF1	10EC1C02	377B0963	ACEFBB7B
8A48243B	F771D47F			

A.3.2.4 Recomputing witness and verifying witness

The witness is recomputed as $\overline{\Pi} = Y^T \cdot S^V \bmod N$.

$\overline{\Pi} = Y^T \cdot S^V \bmod N = R$.

The recomputed witness is the same as the retrieved witness R.

A.4 Numerical example of identity-based signatures giving recovery of the hash-code of message, described in clause 11

In this example, the hash-functions in use are fixed and known to all entities of a domain and hence the hash identifiers are not required. The hash-token is produced by using SHA-1.

The domain parameters, the private signature key and the verification key are the same as A.1.

A.4.1 Signature process

A.4.1.1 Producing pre-signature

The signing entity generates a randomizer, which is a random or pseudo-random integer K with $0 < K < N$. This example uses a 1024-bit randomizer K and the corresponding pre-signature Π is given in A.2.1.1.

A.4.1.2 Preparing message

In this example, the message is same as one in A.2.1.2 and its hash-token H is the same as one in A.3.1.2.

H = hash-token of M

A.4.1.3 Computing witness and the first part of the signature

In this example, the witness forms the first part R of the signature.

$R = \Pi \cdot H \bmod N =$

```
425DCEDD  1D408F3F  6633CEFE  225B92DE  920BE1AF
BD2CE776  446410E7  A08527BC  5ADA0DDD  13C8B371
053FD48C  69BD86FA  1114EE0C  698B7E10  1662529C
9B53662D  64BCD974  4DCF5276  BF576154  407AB43F
85BCC21B  2075B492  142A5724  464A6E30  21777BD0
C5BBC02F  7B93F42C  07916DA7  1BD4D276  81D21D5A
58F5A5AC  D8C3A8EA
```

A.4.1.4 Computing assignment

In this example, the assignment T is equal to R.

$T = R$

A.4.1.5 Computing the second part of the signature

The second part S of the signature is computed as:

$S = K \cdot X^T \bmod N =$

```
3205E60E  84EB7AC6  28866B9D  2CA2A080  D182BF95
27FE80D3  EC2EF3F9  F27F0FF3  4DCEC086  BDB0E072
08EDE468  E5C2F36C  25452213  A3AD3731  3A518CF9
1DE84C4D  25AA2AB9  84D76885  CA9B8A2C  DB388F1D
139AA00F  15340501  D6B1F867  D5313E2B  60F64E34
F7650FED  23C032EB  DDB82BD0  A5AB49FA  CC5F5AAB
AB17F14A  180E6A76
```

The signature Σ is mapped to (R, S).

A.4.2 The verification process

The verifier has the values of the following data items available:

$N, V, Y, M, \Sigma = (R, S)$.

A.4.2.1 Preparing message

H = hash-token of M

A.4.2.2 Retrieving assignment

In this example, the assignment T is equal to R.

$T = R$

A.4.2.3 Recomputing pre-signature

The verifier produces a recomputed value $\overline{\Pi} = Y^T \cdot S^V \bmod N$ of the pre-signature.

If ($Y^T \bmod N$) and ($S^V \bmod N$) are calculated separately, they are given as:

$Y^T \bmod N =$

```
E7D8C6CC  45A8B29F  200F5C52  148824E0  771624FC
DE0FDD70  BC83BE2E  22AAFE20  C0233C14  E6E4E8E2
BEEE7C5C  022FD464  92D0A055  5A0D7782  7AFBEFB4
6D8085FD  AF8A27C1  775285B7  DD7894F4  F05DAF37
BFCF170C  B3638CE7  A796D639  B296D8D9  09B8DE8F
8AC5A98F  3BBDA623  3BB66C95  5127A854  4DC7206B
ED0EB25D  0AB17CDC
```

$S^V \bmod N =$

```
F673DE2   802FA537  DE498937  F969AA84  9F9CE1F0
708BDEB3  D389CD31  2E5BDBD1  C036BDD4  449D4FB4
00AD359F  02B997A6  273B23CF  1F414C1F  C27B7556
74D64887  1B2A3185  12E098D9  2ED260C8  CC511679
886C4EA2  2D638A5A  B727D984  B6F1C2E4  501761BA
74066387  CB0DBA34  A9822A32  E1761CE8  4622E52D
71AB16D9  267B748E
```

A.4.2.4 Recomputing witness and verifying witness

The witness is recomputed as:

$\overline{\Pi} = Y^T \cdot S^V \bmod N = \overline{\Pi} = Y^T \cdot S^V \bmod N \cdot H \bmod N = R.$

The recomputed witness $\overline{\Pi} = Y^T \cdot S^V \bmod N$ is the same as the retrieved witness R.

Given a recomputed value $\overline{\Pi}$ of the pre-signature, the verifier obtains a recomputed value $\overline{H}$ of the witness by computing as:

$\overline{H} = \overline{\Pi}^{-1} R \bmod N =$

```
43F85D2C  D63CD833  94D74837  591623A2  26734169
5BF58243  070B14C7  8B93E68F  4FFF6D7E  FA150C1D
052ECB5F  D743721D  2481469B  C1375059  3A05C239
98E9FA85  3EE7634E  013726BF  8A04FDD3  A9DEA82D
2B93F990  55E842BE  A40DCB27  0E74F8BC  1A243E46
A274BD40  7B5B7A40  45B5F936  B30CAA2E  7C4F1F72
711EA20E  E3CEEFD2
```

$\overline{H} = \overline{\Pi}^{-1} R \bmod N = \overline{\Pi}^{-1} R \bmod \cdot \overline{\Pi} = Y^T \cdot S^V \bmod N$