INTERNATIONAL STANDARD

ISO/IEC 9796-3

First edition
2000-04-15

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

Part 3: Discrete logarithm based mechanisms

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

Partie 3: Mécanismes basés sur les logarithmes discrets

Reference number
ISO/IEC 9796-3:2000(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 9796 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

This first edition cancels and replaces ISO/IEC 9796:1991, which has been technically revised.

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

— *Part 2: Mechanisms using a hash-function*

— *Part 3: Discrete logarithm based mechanisms*

Annex A forms a normative part of this part of ISO/IEC 9796. Annexes B to D are for information only.

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 verification key and not the signature key it is computationally infeasible to produce any message and a valid signature for this message.

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

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

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

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

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

— A process using the signature key; called the signature process.

— A process using the verification key; called the verification process.

There are two types of digital signature mechanisms:

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

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” (see ISO/IEC 9796).

NOTE Any signature mechanism giving message recovery, for example, the mechanisms specified in 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.

The mechanisms specified in ISO/IEC 9796 give either total or partial recovery, aiming at reducing storage and transmission overhead.

The mechanisms specified in this part of ISO/IEC 9796 use a hash-function for hashing the entire message. ISO/IEC 10118 specifies hash-functions for digital signatures. If the message is short enough, then the entire message can be included in the signature, and recovered from the signature in the verification process. Otherwise, a part of the message can be included in the signature and the rest of it is stored and/or transmitted along with the signature.

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 two randomized digital signature schemes giving message recovery. The security of both schemes is based on the difficulty of the discrete logarithm problem. The first scheme is defined on a prime field and the second one on an elliptic curve.

This part of ISO/IEC 9796 also defines a redundancy scheme using hash-codes and specifies how the basic signature schemes are to be combined with the redundancy scheme.

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

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9796. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 9796 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

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

ISO/IEC 11770-3:1999, *Information technology – Security techniques – Key management – Part 3: Mechanisms using asymmetric techniques*.

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

ISO/IEC 15946 (parts 1 and 2, to be published), *Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 1: General* and *Part 2: Digital signatures*.

3 Terms and definitions

For the purposes of this part of ISO/IEC 9796, the following definitions apply.

3.1 assignment
[ISO/IEC 14888-1] A data item which is a function of the witness and possibly of a part of the message, and forms part of the input to the signature function.

3.2 certification authority
[ISO/IEC 11770-3] A centre trusted to create and assign public key certificates. Optionally, the certification authority may create and assign keys to the entities.

3.3 collision-resistant hash-function
[ISO/IEC 10118-1] A hash-function satisfying the following property:

— it is computationally infeasible to find any two distinct inputs which map to the same output.

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

3.4 data input
A data item which depends on the entire message and forms a part of the input to the signature function.

3.5 domain parameter
[ISO/IEC14888-1] A data item which is common to and known by or accessible to all entities within the domain.

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

3.6 hash-code
[ISO/IEC 10118-1] The string of bits which is the output of a hash-function.

3.7 hash-function
[ISO/IEC 10118-1] A function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties:

— for a given output, it is computationally infeasible to find an input which maps to this output; and

— for a given input, it is computationally infeasible to find a second input which maps to the same output.

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

3.8 hash-token
[ISO/IEC 14888-1] A concatenation of a hash-code and an optional control field, which can be used to identify the hash-function and the padding method.

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

3.9 message
A string of bits of any length.

3.10 pre-signature
[ISO/IEC 14888-1] A value computed in the signature process which is a function of the randomizer but is independent of the message.

3.11 public key certificate
[ISO/IEC 11770-3] The public key information of an entity signed by the certification authority and thereby rendered unforgeable.

NOTE In the context of this part of ISO/IEC 9796 the public key information contains the information about the verification key and the domain parameters.

3.12 randomized
[ISO/IEC 14888-1] Dependent on a randomizer.

3.13 randomizer
[ISO/IEC 14888-1] A secret data item produced by the signing entity in the pre-signature production process, and not predictable by other entities.

3.14 signature
The string of bits resulting from the signature process.

NOTE This string of bits may have internal structure specific to the signature mechanism. The signatures produced by the mechanisms specified in this part of ISO/IEC 9796 have two parts, of which only the second one depends on the signature key.

3.15 signature function
[ISO/IEC 14888-1] A function in the signature process which is determined by the signature key and the domain parameters. A signature function takes the assignment and possibly the randomizer as inputs and gives the second part of the signature as output.

NOTE In the context of this part of ISO/IEC 9796, the assignment is the data input.

3.16 signature key
[ISO/IEC 14888-1] A secret data item specific to an entity and usable only by this entity in the signature process.

3.17 signature process
[ISO/IEC 14888-1] A process which takes as inputs the message, the signature key and the domain parameters, and which gives as output the signature.

3.18 signed message
[ISO/IEC 14888-1] A 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.

3.19 verification function
[ISO/IEC 14888-1] A function in the verification process which is determined by the verification key and which gives a recomputed value of the witness as output.

3.20 verification key
[ISO/IEC 14888-1] A data item which is mathematically related to an entity's signature key and which is used by the verifier in the verification process.

3.21 verification process
[ISO/IEC 14888-1] A process which takes as input the signed message, the verification key and the domain parameters, and which gives as output the result of the signature verification: valid or invalid.

3.22 witness
[ISO/IEC 14888-1] A data item which provides evidence to the verifier.

NOTE In the context of this part of ISO/IEC 9796 the witness is based on a hash-token.

4 Symbols, conventions, and legend for figures

4.1 Symbols and notation

The following symbols and notation are used in this part of ISO/IEC 9796.

D, D'	data input, recovered data input, respectively
F	finite field
G	element of a prime field or point on an elliptic curve
H, H', H''	hash-token, recovered (truncated) hash-token, recomputed (truncated) hash-token, respectively
$\mathfrak{I}$	elliptic curve

K	randomizer
k, m, n	positive integers
L	length in bytes of (truncated) hash-token
L_F	size of the field F
L_{rec}, L_{clr}, L_P, L_Q	length in bytes of M_{rec}, M_{clr}, P, and Q, respectively
L_1, L_2	length in bytes of short and long redundancy, respectively
len_Q	length in bits of Q
M, M_{clr}, M_{rec}	message, nonrecoverable part of message, and recoverable part of message, respectively
M', M'_{rec}	recovered message, recovered part of message, respectively
NA	Nth multiple of a point A on an elliptic curve
O	point at infinity on an elliptic curve
P	prime number
Π, Π'	pre-signature, recovered pre-signature, respectively
Q	prime number, order of G
R	first part of the signature
S	second part of the signature
x	x-coordinate of a point on an elliptic curve
X	signature key
y	y-coordinate of a point on an elliptic curve
Y	verification key
$A * B$	sum of points A and B on an elliptic curve
U mod V	the remainder, when integer U is divided by integer V
$U \equiv V \pmod{W}$	integer U is congruent to integer V modulo integer W

4.2 Coding convention, length and size of the field

All integers are written with the most significant digit (or bit, or byte) in the leftmost position.

Given a non-negative integer n and an integer U in the range $0 \le U < 2^n$, the integer U is converted to a string of bits of length n using its binary representation as shown below:

$$U = u_1 \cdot 2^{n-1} + u_2 \cdot 2^{n-2} + \ldots + u_{n-1} \cdot 2 + u_n \rightarrow (u_1, u_2, \ldots, u_{n-1}, u_n).$$

Conversely, a string of bits of length n is converted to an integer by the rule

$$(u_1, u_2, \ldots, u_{n-1}, u_n) \rightarrow u_1 \cdot 2^{n-1} + u_2 \cdot 2^{n-2} + \ldots + u_{n-1} \cdot 2 + u_n.$$

If an integer U is in the range $2^{k-1} \leq U < 2^k$, it is said that the length of U in bits is equal to k, and the notation $k = len_U$ is used.

If an integer U is in the range $256^{m-1} \leq U < 256^m$, it is said that the length of U in bytes equals m, and the notation $m = L_U$ is used. Hence, L_U is the least integer with the property $8 \cdot L_U \geq len_U$.

If F is a field over a prime P, the field size L_F is defined as $L_F = L_P$. If F is a field over a power of two 2^n, the field size L_F is defined as the least integer with the property $8 \cdot L_F \geq n$.

4.3 Legend for figures

The legend for figures in this part of ISO/IEC 9796 is as follows:

step of the process

mandatory data

optional data

5 Requirements

Users who wish to employ a digital signature mechanism from this part of ISO/IEC 9796 shall select:

— a signature scheme from between the two specified in Clause 9 and Clause 10;

— size L_F of the underlying field F;

— parameter Q for the selected signature scheme;

— the byte length L_1 of the short redundancy and the byte length L_2 of the long redundancy, where $8L_1$ and $8L_2$ are strictly less than the length len_Q of the binary representation of Q, and

— a collision-resistant hash-function which produces hash-codes such that the length of the resulting hash-tokens in bytes is L_2 .

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

Short redundancy is used if the entire message is recoverable from the signature. Then the bit-length of the message is at most $len_Q - 8L_1 - 1$ bits.

Long redundancy is used if a part of the message is not recoverable from the signature. Then the recoverable part of the message is at most $len_Q - 8L_2 - 1$ bits.

NOTE The sizes of the parameters Q, L_1 and L_2 also affect the security level of the signatures giving message recovery. Typical values of L_1 are 8 or 10. Typical values of L_2 vary from 17 to 21. It is also possible to set $L_1 = L_2$.

5.1 Options for binding signature mechanism and hash-function

When a digital signature mechanism uses a hash-function, there shall be a binding between the signature mechanism and the hash-function in use. Without such a binding, an adversary may claim the use of a weak hash-function (and not the actual one) and thereby forge the signature. There are various ways to accomplish the required binding. The following options are presented 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 SHA-1.

b) Allow a set of hash-functions and explicitly indicate in every signature the hash-function in use by a hash-function identifier included as part of the signature calculation. The hash-function identifier is an extension of the hash-code: it indicates how to derive the hash-code. The verification process shall exclusively use the hash-function indicated by the identifier in the signature. This part of ISO/IEC 9796 and ISO/IEC 9796-2 give examples of this option.

c) Allow a set of hash-functions and explicitly indicate the hash-function in use in the certificate domain parameters. Within the domain, the verification process shall exclusively use the hash-function indicated in the certificate. Outside the domain, there is a risk due to less rigorous certification authorities. If other certificates may be created, then other signatures may be created. Then the attacked user would be in a dispute situation with the certification authority that produced the other certificate.

d) 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 method. However, there is a risk that an adversary may forge a signature using another hash-function.

The user of a digital signature mechanism should conduct a risk assessment considering the costs and benefits of the various alternatives. This assessment includes the cost associated with the possibility of a bogus signature being produced.

6 Signature process

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

— producing a randomizer and the pre-signature

— splitting the message

— hashing the message

— formatting the data input

— computing the signature

— formatting the signed message

6.1 Producing the pre-signature

Pre-signature is an intermediate data item produced in any randomized signature mechanism. First a randomizer K, which is an integer, is produced. Then the pre-signature Π is computed as an element in the finite field specified by the signature scheme, see 9.3 or 10.3, respectively. The pre-signature is a public data item, while the value of the randomizer shall be usable only by the signature process.

Randomizers can be produced and corresponding pre-signatures computed off-line and stored securely for future use by the signature process.

6.2 Producing the hash-token

The input to the hash-function is formed by concatenating the following strings of bits from the left to the right

— the 64-bit string representing the integer L_{rec};

— the 64-bit string representing the integer L_{clr};

— the $8\cdot(L_{clr} + L_{rec})$-bit string of M, and

— the pre-signature Π converted to a string of bits of length $8\cdot L_F$.

The output from the hash-function is the hash-code. The hash-token H is formed by the hash-code by itself, or with the hash-function identifier concatenated to the right of the hash-code. The choice of whether or not the hash-token includes the hash-function identifier shall be controlled by the domain parameters.

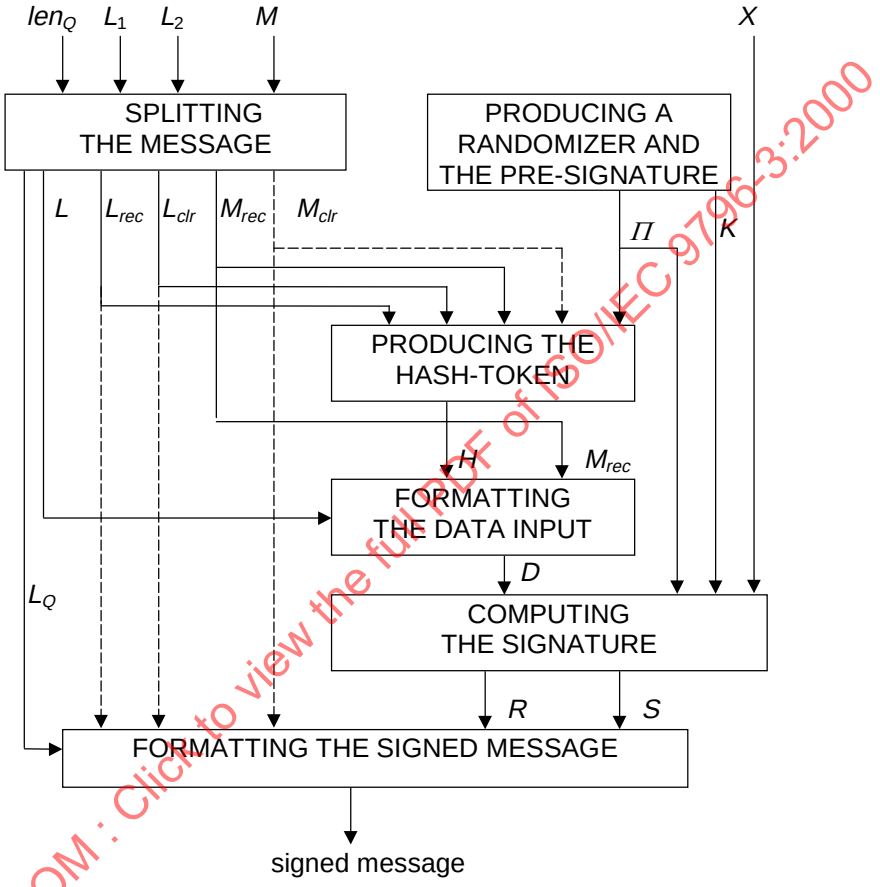

Figure 1 — Signature process

6.3 Formatting the data input

The data input D is formed by concatenating the following two strings of bits from the left to the right:

— the L leftmost bytes of H, and

— the L_{rec} bytes of M_{rec}.

Then $8\cdot(L + L_{rec}) \leq len_Q - 1$. Consequently, the data input D, after conversion to an integer, is in the range $0 \leq D < Q$.

6.4 Computing the signature

The computation of the signature is specified by the selection of the signature scheme from between the two schemes specified in Clause 9 or Clause 10 and shall be indicated by the domain parameters.

The signatures produced by the schemes specified in this part of ISO/IEC 9796 consist of two parts R and S. The first part R is computed as a function of the data input D and the pre-signature Π, see 9.4 or 10.4, respectively. The second part S is computed as a function of the first part R, the randomizer K, and the signature key X, see 9.5 or 10.5, respectively.

6.5 Formatting the signed message

Knowledge of the lengths of the recoverable and non-recoverable parts of the message is necessary for the successful opening and verification of the signed message. Unless given by the domain parameters, this information must be included in the signed message.

The signed message is a concatenation from the left to the right of the following data items:

— the 64-bit string representing the integer L_{rec} (optional);

— the 64-bit string representing the integer L_{clr} (optional);

— text, an optional data field of arbitrary length, whose use is outside the scope of this part of ISO/IEC 9796;

— the $8 \cdot L_{clr}$-bit string of the non-recoverable part M_{clr} of the message (if $L_{clr} = 0$, then M_{clr} does not exist);

— the $8 \cdot L_Q$-bit string of the first part R of the signature; and

— the $8 \cdot L_Q$-bit string of the second part S of the signature.

7 Verification process

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

— opening the signed message

— recovering the pre-signature and the data input

— recovering the message and the hash-token

— recomputing (and truncating) the hash-token

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

7.1 Opening the signed message

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

— the lengths of the different message parts included in the signed message; and

— the values of the parameters L_Q, L_1 and L_2.

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

— the $8 \cdot L_Q$ rightmost bits, which form the second part S of the signature;

— from the remaining string, the $8 \cdot L_Q$ rightmost bits, which form the first part R of the signature;

— from the remaining string, the 64 bit leftmost bits representing the integer L_{rec}, if it is included in the signed message;

— the leftmost 64 bits representing the integer L_{clr}, if it is included in the signed message;

— from the remaining string, the $8 \cdot L_{clr}$ rightmost bits, which form the non-recoverable part M_{clr} of the message (if $L_{clr} = 0$, then M_{clr} does not exist);

— the remaining string is the text, which is an optional data field of arbitrary length, whose use is outside the scope of this part of ISO/IEC 9796.

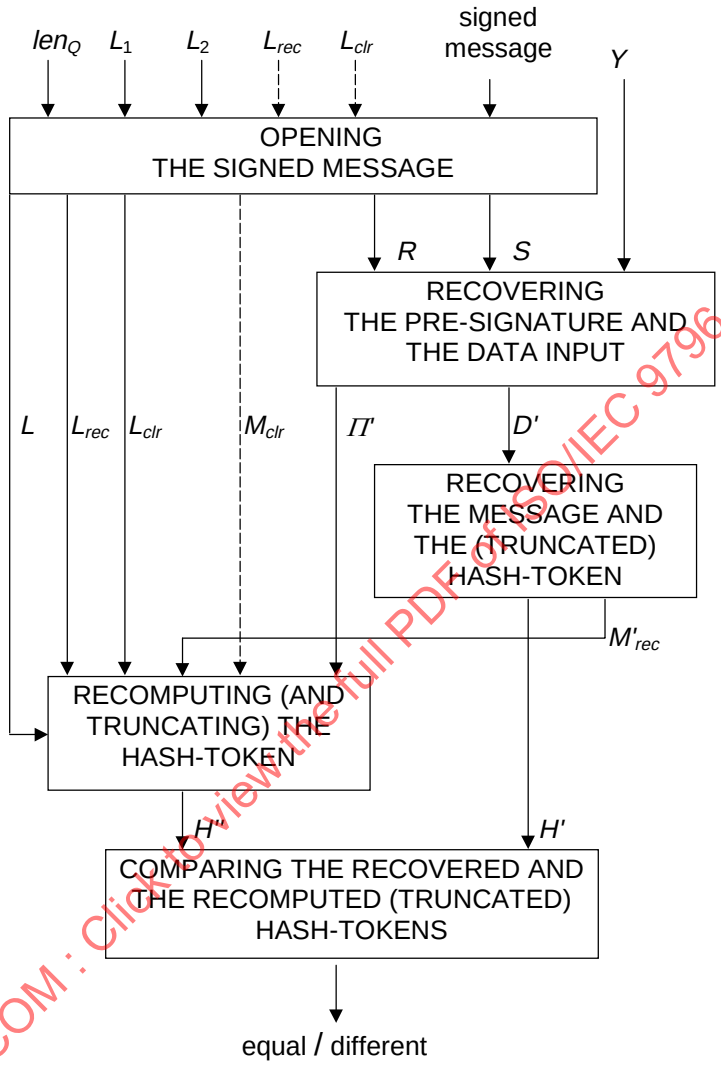

Figure 2 — Verification process

The verifier ensures that

— R is not the all zero string;

— if $L_{clr} = 0$, then $8 \cdot L_{rec} + 8 \cdot L_1 \leq len_Q - 1$, and

— if $L_{clr} \neq 0$, then $8 \cdot L_{rec} + 8 \cdot L_2 \leq len_Q - 1$.

If any of these three checks fail then the signature shall be rejected. Further, the verifier sets $L = L_1$, if $L_{clr} = 0$. Otherwise, $L = L_2$.

7.2 Recovering the pre-signature and the data input

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

— the public parameters which specify the signature scheme in use; and specifically,

— the verification key Y of the signing entity.

The computations at this step are specific to the signature scheme in use. The pre-signature and the data input is recovered from the signature as specified in 9.6-7, or 10.6-7, respectively. The recovered pre-signature is Π' and the recovered data input is D'.

7.3 Recovering the message and the (truncated) hash-token

The recovered data input D' is converted to a string of bits of length len_Q. Then two data items, the recovered hash-token H' and the recovered part M'_{rec} of the message are extracted from D' as follows:

— the L leftmost bytes form H', and

— from the remaining string the L_{rec} leftmost bytes form M'_{rec}.

The entire recovered message M' is obtained by concatenating M'_{rec} and M_{clr}.

7.4 Recomputing the hash-token

First, the hash-function used by the signing entity in 6.3 is identified, possibly by retrieving the hash-function identifier from the recovered hash-token. Then the hash-code is recomputed by hashing the data string which is obtained by concatenating from the left to the right the following data items:

— the 64-bit string representing the integer L_{rec};

— the 64-bit string representing the integer L_{clr};

— the $L_M = L_{clr} + L_{rec}$ byte string of M', and

— the recovered pre-signature Π', converted from a field element to a string of bits.

The recomputed hash-code is used to obtain the recomputed hash-token by optionally concatenating the hash-function identifier. The L leftmost bytes of the resulting hash-token are extracted to form the recomputed (truncated) hash-token H''.

7.5 Comparing the recovered and the recomputed (truncated) hash-tokens

The final step in the verification process is to compare the recomputed (truncated) hash-token H'' with the recovered (truncated) hash-token H'. The signature shall be rejected if these two values are not equal.

8 Signature schemes giving message recovery

In this part of ISO/IEC 9796 two signature schemes are defined. A signature scheme consists of the following functions and procedures:

— producing domain parameters

— producing signature and verification key

— producing randomizer and pre-signature

— computing the first part of the signature

— computing the signature function

— computing verification function

— recovering the data input

The signature schemes specified in this standard give message recovery. More precisely, the data which is input to the signature function is recovered from the signature using the verification function.

9 Signature scheme on a prime field

9.1 Domain parameters

The domain parameters of the digital signature scheme are the following:

— a prime number Q,

— a prime number P such that Q divides $P - 1$, and

— an integer G, such that $1 < G < P - 1$ and $G^Q \equiv 1 \pmod{P}$.

NOTE The set of domain parameters may also contain a seed and a method for generating P and Q.

Prior to use of the domain parameters a user shall have assurance about their validity. A routine for the validation of domain parameters is given in Annex A.

9.2 Signature and verification key

The signature key is an integer X, such that $1 \leq X < Q$. The signing entity shall keep the signature key secret. The verification key is an integer Y, such that $1 < Y < P$ and that $Y \equiv G^X \pmod{P}$.

Prior to use of the verification key the verifier shall have assurance about its validity and ownership. A routine for the validation of the verification key is given in Annex A.

9.3 Randomizer and pre-signature

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

— The used randomizers are never disclosed, since knowledge of a randomizer and the signature generated using this randomizer can be used to compute the private signature key.

— The randomizers are statistically unique, that is, the probability that a randomizer is used twice is negligble. If the same randomizer is used to produce signatures for two different messages, then the signature key can be computed from the signatures.

The pre-signature is computed as a function of the randomizer. The pre-signature is an integer Π, such that $1 < \Pi < P$, and $\Pi \equiv G^K \pmod{P}$.

9.4 The first part of the signature

The first part of the signature is computed as a function of the data input D and the pre-signature Π. The first part of the signature is an integer R such that $0 < R < Q$ and $R \equiv \Pi + D \pmod{Q}$. If for given data D and a produced pre-

signature Π the sum $D + \Pi$ is divisible by Q, then a new value of the randomizer must be chosen and a new value of pre-signature computed.

9.5 Signature function

The signature function is determined by the signature key X. Given the first part of the signature R and the randomizer K, the second part of the signature is an integer S such that $0 \leq S < Q$ and $S \equiv K - XR \pmod{Q}$.

9.6 Verification function

The verification function is determined by the verification key Y. Given the signature (R,S) the pre-signature is recovered by computing $\Pi' = G^S Y^R \bmod P$.

9.7 Recovering the data input

Given the first part R of the signature and the recovered pre-signature Π' the data D is recovered by computing $D' = (R - \Pi') \bmod Q$.

10 Signature scheme on an elliptic curve

10.1 Domain parameters

The domain parameters of the digital signature scheme are the following:

— a finite field F;

— a prime number Q;

— an elliptic curve $\Im$ over F which has a unique cyclic subgroup of order Q, and

— a point G on $\Im$ of order Q.

An elliptic curve $\Im$ over a field F is given by a pair (a,b) of elements in F, which define the equation of $\Im$ in its standard form unless $4a^3 + 27b^2 = 0$ in F. A point on $\Im$ is either a point at infinity, which is denoted by O, or it has two coordinates, the x-coordinate and the y-coordinate, which are elements of F and satisfy the bivariate equation of $\Im$. The points on $\Im$ form an abelian group under a group law denoted by the symbol '$*$', for which O is the neutral element. Given an integer N and a point A on $\Im$, the Nth multiple $A * A * \ldots * A$ of A is denoted by NA. When computing the addition on $\Im$ the coordinates of the points are considered as elements in F. The users shall also agree on a common procedure to convert the elements of F to non-negative integers.

Prior to use of the domain parameters a user shall have assurance about their validity. A routine for the validation of domain parameters is given in Annex A.

The equations and group law of elliptic curves are different depending on the type of the underlying field. In this document the definitions are given for a finite field over a prime and for a finite field over a power of two.

10.1.1 Equation and group law for a field over a prime

Let F be a finite field over a prime, which is larger than three. An elliptic curve $\Im$ over F is described by a "short Weierstrass equation", which is an equation of the form $y^2 = x^3 + a\,x + b$.

The group law is defined as follows. Let $A = (x_1,y_1)$ and $B = (x_2,y_2)$ be two points that are different from the point at infinity on $\Im$. If $x_1 = x_2$ and $y_1 + y_2 = 0$ in F, it is defined that $A * B = O$. Then A is called the inverse of B, or vice versa. Assume now that A is not the inverse of B. Then $A * B = (x_3,y_3)$, where $x_3 = r^2 - x_1 - x_2$, $y_3 = r(x_1 - x_3) - y_1$, and either $r = (y_2 - y_1)/(x_2 - x_1)$, if $A \neq B$, or $r = (3x_1^2 + a)/2y_1$, if $A = B$.

10.1.2 Equation and group law for a field over a power of two

If F is a field over a power of 2, then an elliptic curve $\mathfrak{I}$ over F is given by an equation of the form $y^2 + xy = x^3 + ax^2 + b$, where $b \neq 0$.

The group law is defined as follows. Let $A = (x_1,y_1)$ and $B = (x_2,y_2)$ be two points that are different from the point at infinity on $\mathfrak{I}$. If $x_1 = x_2$ and $y_1 + y_2 = x_1$ in F, it is defined that $A * B = O$. Then A is called the inverse of B, or vice versa. Assume now that A is not the inverse of B. Then there are two cases, either $A \neq B$ or $A = B$. If $A \neq B$, then $A * B = (x_3,y_3)$, where $x_3 = r^2 + r + x_1 + x_2 + a$, $y_3 = r(x_1 + x_3) + x_3 + y_1$, and $r = (y_2 + y_1)/(x_2 + x_1)$. If $A = B$, then $A * B = (x_3,y_3)$, where $x_3 = r^2 + r + a$, $y_3 = x_1^2 + (r + 1)x_3$, and $r = x_1 + (y_1/x_1)$.

10.2 Signature and verification key

The signature key is an integer X, such that $1 \leq X < Q$. The signing entity shall keep the signature key secret. The verification key is a point Y on $\mathfrak{I}$ such that $Y = XG$.

Prior to use of the verification key the verifier shall have assurance about its validity and ownership. A routine for the validation of the verification key is given in Annex A.

10.3 Randomizer and pre-signature

Prior to each signature computation the signing entity must have a fresh, secret value of randomizer available. The randomizer is an integer K such that $1 \leq K < Q$. The requirements to the implementation of the signature scheme concerning randomizers are the same as those given in 9.3.

The pre-signature is computed as a function of the randomizer. The pre-signature Π is the x-coordinate of the point KG converted to an integer.

10.4 Computing the first part of the signature

The first part of the signature is computed as a function of the pre-signature Π and the data input D, which is an integer such that $0 \leq D < Q$. The first part of the signature is an integer R such that $0 < R < Q$ and $R \equiv \Pi + D \pmod{Q}$. If for given data D and a produced pre-signature Π the sum $\Pi + D$ is divisible by Q, then a new value of randomizer must be chosen and a new pre-signature computed.

10.5 Signature function

The signature function is determined by the signature key X. Given the first part of the signature R and the randomizer K the second part of the signature is an integer S such that $0 \leq S < Q$ and $S \equiv K - XR \pmod{Q}$.

10.6 Verification function

The verification function is determined by the verification key Y. Given the signature (R,S) the pre-signature Π is recovered by computing the x-coordinate Π' of the point $SG * RY$ and by converting the resulting field element to an integer.

10.7 Recovering the data input

Given the first part R of the signature and the recovered pre-signature Π' the data input D is recovered by computing $D' = (R - \Pi') \bmod Q$.

Annex A
(normative)

Validation of domain parameters and public keys

In this annex optional routines for validation of domain parameters and verification keys are specified.

A.1 Signature scheme on a prime field

A.1.1 Domain parameter validation

The primes *P* and *Q* must be generated in an agreed upon manner (for example, by using a seeded hash-function as found in the specification of DSA [ISO/IEC 14888-3]) in order to thwart attacks where *P* and *Q* are chosen to have specific rare properties that allow attacks.

Given a candidate set of the domain parameters: *seed, P, Q* and *G*, they can be validated for arithmetic conformance with this standard by doing the following:

a) Ensure that *seed* generates *P* and *Q* according to an agreed upon method.

b) Ensure that *P* is a prime or has passed an agreed upon probabilistic primality test.

c) Ensure that *Q* is a prime or has passed an agreed upon probabilistic primality test.

d) Ensure that *Q* divides *P* - 1, that is, $P - 1 = QH$, for some integer *H*, and that *Q* does not divide *H*.

e) Ensure that *G* is in the correct range, that is, $1 < G < P - 1$.

f) Ensure that *G* has correct order *Q*, that is, $G^Q \equiv 1 \pmod{P}$.

A.1.2 Verification key validation

Given that the domain parameters have been validated, a candidate verification key *Y* may be validated for arithmetic conformance with this standard by doing the following:

a) Ensure that *Y* is in the correct range, that is, $1 < Y < P$.

b) Ensure that *Y* has the correct order, that is $Y^Q \equiv 1 \pmod{P}$.

A.2 Signature scheme on an elliptic curve

A.2.1 Domain parameter validation

There are no known attacks that require the generation of elliptic curve parameters in an agreed upon manner via the use of a seeded hash-function.

NOTE However, a generation method using a seeded hash-function is recommended and can be found in [1] and [2], see Bibliography.

Given a candidate set for domain parameters, *seed* (optional), *F*, (*a*,*b*), and a point *G* of prime order *Q*, they may be validated for arithmetic conformance with this standard by doing the following:

a) Ensure that F specifies a prime power, that is, F is actually a field. Typically, F is a field over a prime P or a power of 2, say 2^m, where m is a positive integer.

b) Ensure that a and b are elements of the field F.

c) Ensure that (a,b) define an elliptic curve. Then one of the following routines is applicable.

   1) If F is over a prime P, ensure that

      i) $0 \le a \le P - 1$;

      ii) $0 \le b \le P - 1$;

      iii) $4\,a^3 + 27\,b^2$ is not an integer multiple of P.

   2) If F is over a power 2^m of 2, ensure that

      i) a is a string of m bits;

      ii) b is a string of m bits;

      iii) $b \neq 0$.

d) Ensure that G is a point on the elliptic curve, that is, $G = (x, y)$, where x and y are elements of the field F and satisfy the defining equation.

   1) Ensure that x and y are elements in F.

   2) If F is a field over a prime, ensure that $y^2 = x^3 + a\,x + b$ in F.

   3) If F is a field over a power of 2, ensure that $y^2 + xy = x^3 + a\,x^2 + b$ in F.

e) Ensure that G has order Q, that is, ensure that G is not the point at infinity and that QG is the point at infinity.

f) Ensure that Q is a prime or has passed an agreed upon probabilistic primality test.

g) Ensure that the MOV condition holds: The MOV condition is verified as follows: Input is a threshold B, a prime power q, and a prime n (where n is a prime divisor of the number of points of the elliptic curve which is defined over the field of order q). Output is "True", if MOV condition is satisfied, "False" otherwise. It is suggested that $B = 20$ is an appropriate value for B. Process as follows:

   1) set $t = 1$.

   2) For $i = 1$ to B do:

      3) Set $t = tq \bmod n$.

      4) If $t = 1$, then output "False", and stop.

   5) End For

   6) Output "True".

NOTE The MOV condition implies that the number of points on the curve is not equal to the number of elements of the underlying field plus one.

h) Ensure that the curve is not anomalous. This is verified by checking that the number of points on the curve is not equal to the number of elements of F.

i) Ensure that Q divides the number of points on the curve, but that Q^2 does not.

A.2.2 Verification key validation

Given that the domain parameters have been validated, a candidate verification key Y may be validated for arithmetic conformance with this standard by doing the following:

a) Ensure that Y is a point on the elliptic curve, that is, $Y = (x,y)$, where x and y are elements of the field F and satisfy the defining equation.

   1) Ensure that x and y are elements in the field F.

   2) If F is a field over a prime, ensure that $y^2 = x^3 + a\,x + b$ in the field.

   3) If F is a field over a power of 2, ensure that $y^2 + xy = x^3 + a\,x^2 + b$ in the field.

b) Ensure that Y has the correct order, that is, Y is not the point at infinity and QY is the point at infinity.

Annex B
(informative)

Numerical examples I — Signature mechanisms on finite fields

Throughout this annex we refer to ASCII coding of data strings; this is equivalent to coding using ISO 646.

B.1 Examples 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
```

len_Q 1023 bits

G 2

Signature key *X*
```
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*
```
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 9e36a339 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
```

B.1.1 Example with hash-function SHA-1 (Dedicated Hash-Function 3 of ISO/IEC 10118-3)

Length of hash-token L 21 bytes

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
```

B.1.2 Example with hash-function RIPEMD-160 (Dedicated Hash-Function 1 of ISO/IEC 10118-3)

Length of hash-token L 21 bytes

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
```

B.1.3 Example with hash-function RIPEMD-128 (Dedicated Hash-Function 2 of ISO/IEC 10118-3)

Length of hash-token L 17 bytes

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
```

B.2 Example with total recovery

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
```

len_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 `478bbe64 7cd50ef3 67ebe30f dc10c9e0 1ce37fb5`

Verification key Y

```
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
```