INTERNATIONAL STANDARD

ISO/IEC 18370-2

First edition
2016-07-01

Information technology — Security techniques — Blind digital signatures —

Part 2: Discrete logarithm based mechanisms

Technologie de l'information — Techniques de sécurité — Signatures numériques en aveugle —

Partie 2: Mécanismes fondés sur le logarithme discret

Reference number
ISO/IEC 18370-2:2016(E)

Contents

Page

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 18370 consists of the following parts, under the general title *Information technology — Security techniques — Blind digital signatures*:

— *Part 1: General*

— *Part 2: Discrete logarithm based mechanisms*

Further parts may follow.

Introduction

Blind digital signature mechanisms are a special type of digital signature mechanism, as specified in ISO/IEC 9796 (all parts) and ISO/IEC 14888, which allow a user (a requestor) to obtain a signature from a signer of the user's choice, without giving the signer any information about the message that is signed or the resulting signature.

In some mechanisms, the signer does not completely lose control over the signed message since the signer can include explicit information in the resulting signature under an agreement with the requestor. These types of blind signatures are called blind signatures with partial disclosure.

Other mechanisms allow a requestor to receive a blind signature on a message not known to the signer but the choice of the message is restricted and needs to conform to certain rules. Such mechanisms are called blind signature mechanisms with selective disclosure.

Depending on the mechanism, it may be possible for an authorized entity to trace a signature to the requestor who requested it. Such an entity can either identify a signature that resulted from a given signature request (signature tracing), or link a signature to the receiver who requested it (requestor tracing). Blind signature mechanisms with tracing features are called traceable blind signature mechanisms.

ISO/IEC 18370 specifies blind digital signature mechanisms, as well as three variants: blind digital signature mechanisms with partial disclosure, blind digital signature mechanisms with selective disclosure and traceable blind digital signature mechanisms. ISO/IEC 18370-1 specifies principles and requirements for these mechanisms. This part of ISO/IEC 18370 specifies several specific instances of these mechanisms.

The security of blind digital signature mechanisms and their variants depends on computational problems believed to be intractable, i.e. problems for which, given current knowledge, finding a solution is computationally infeasible, such as the integer factorization problem or the discrete logarithm problem in an appropriate group. The mechanisms specified in this part of ISO/IEC 18370 are based on the latter problem.

ISO/IEC 18370 does not specify mechanisms for key management or for certification of public keys. A variety of means are available for obtaining a reliable copy of the public verification key, e.g. a public key certificate. Techniques for managing keys and certificates are outside the scope of ISO/IEC 18370. For further information, see ISO/IEC 9594-8, ISO/IEC 11770-3 and ISO/IEC 15945.

This part of ISO/IEC 18370 specifies mechanisms that use a collision resistant hash-function to hash the message to be blindly signed. ISO/IEC 10118 specifies hash-functions.

The generation of key pairs requires random bits and prime numbers. The generation of signatures requires random bits. Techniques for producing random bits and prime numbers are outside the scope of ISO/IEC 18370. For further information, see ISO/IEC 18031 and ISO/IEC 18032.

Information technology — Security techniques — Blind digital signatures —

Part 2: Discrete logarithm based mechanisms

1 Scope

This part of ISO/IEC 18370 specifies blind digital signature mechanisms, together with mechanisms for three variants of blind digital signatures. The variants are blind digital signature mechanisms with partial disclosure, blind digital signature mechanisms with selective disclosure and traceable blind digital signature mechanisms. The security of all the mechanisms in this part of ISO/IEC 18370 is based on the discrete logarithm problem.

For each mechanism, this part of ISO/IEC 18370 specifies the following:

— the process for generating the keys of the entities involved in these mechanisms;

— the process for producing blind signatures;

— the process for verifying signatures.

This part of ISO/IEC 18370 specifies another process specific to blind signature mechanisms with selective disclosure, namely, the following:

— the presentation process.

Furthermore, this part of ISO/IEC 18370 specifies other processes specific to traceable blind signature mechanisms, namely, the following:

a) the process for tracing requestors;

b) the process for tracing signatures;

c) the requestor tracing evidence evaluation process (optional);

d) the signature tracing evidence evaluation process (optional).

2 Normative references

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

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

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 18370-1 and the following apply.

3.1
abelian group
group $(G, *)$ such that $a * b = b * a$ for every a and b in G

3.2
cyclic group
group G of n elements that contains an element a in G, called the generator, of order n

[SOURCE: ISO/IEC 14888-3:2006, 3.2]

3.3
elliptic curve over a finite field
set E of points $P = (x, y)$, where x and y are elements of the *finite field* (3.6), that satisfy a certain equation, together with an extra point referred to as the point at infinity

Note 1 to entry: In this part of ISO/IEC 18370, only finite fields containing exactly q elements for a prime $q > 3$ are considered. In this case, the equation that every point $P = (x, y)$ of E (other than the point at infinity) should satisfy is of the form $y^2 = x^3 + ax + b$. The finite field elements a and b should satisfy $4a^3 + 27b^2 \neq 0_F$ (where 0_F is the additive identity element of the finite field).

Note 2 to entry: The set of points E, together with an appropriately defined operation, forms a *finite commutative group* (3.5), where the point at infinity is the identity element.

3.4
field
set of elements S and a pair of operations (+,*) defined on S, such that: i) $a * (b + c) = a * b + a * c$ for every a, b and c in S, ii) S together with + forms an *abelian group* (3.1) (with identity element 0), and iii) S excluding 0 together with * forms an abelian group

3.5
finite commutative group
abelian group (3.1) $(G, *)$ with a finite number of elements

Note 1 to entry: If $a^0 = e$, and $a^{n+1} = a * a^n$ (for $n \geq 0$) is defined recursively, the order of $a \in G$ is the least positive integer n, such that $a^n = e$.

Note 2 to entry: In some cases, such as when G is the set of points on an elliptic curve, arithmetic in the finite set G is described using additive notation.

3.6
finite field
field (3.4) such that the underlying set of elements is finite

Note 1 to entry: For any positive integer, m and a prime p, there exists a finite field containing exactly $q = p^m$ elements. This field is unique up to an isomorphism and is denoted by F_q.

[SOURCE: ISO/IEC 18033-2:2006, 3.21]

3.7
group
set of elements G and an operation * defined on the set of elements such that: i) $(a * b) * c = a * (b * c)$ for every a, b and c in G, ii) there exists an identity element, e in G, such that $a * e = e * a = a$ for every a in G, and iii) for every a in G, there exists an inverse element, a^{-1} in G, such that $a * a^{-1} = a^{-1} * a = e$

3.8
security parameters
variables that determine the security strength of a mechanism

4 Symbols

$a \in A$	indicates that element a is in set A
$a \parallel b$	concatenation of data items a and b in the order specified. In cases where the result of concatenating two or more data items is input to a cryptographic algorithm as part of one of the mechanisms specified in this part of ISO/IEC 18370, this result shall be composed so that it can be uniquely resolved into its constituent data strings, i.e. so that there is no possibility of ambiguity in interpretation. This latter property could be achieved in a variety of different ways, depending on the application. For example, it could be guaranteed by a) fixing the length of each of the substrings throughout the domain of use of the mechanism, or b) encoding the sequence of concatenated strings using a method that guarantees unique decoding, e.g. using the distinguished encoding rules defined in ISO/IEC 8825-1[1].
$A \subseteq B$	indicates that the set A is a subset of or equal to set B
$A \setminus B$	when A and B are sets, this represents the set of elements present in A but not in B.
$\|D\|$	bit length of D if D is a bit string, or bit size of D if D is a non-negative number (i.e. 0 if $D = 0$, or the unique integer i such that $2^{i-1} \le D < 2^i$ if $D > 0$).
E	an elliptic curve over the finite field F_p, for a prime $p > 3$
$E(F_p)$	the set of all points (x, y), $x \in F_p$, $y \in F_p$, which satisfy the defining equation of the curve E, together with the point at infinity, O_E
$\#E(F_p)$	the order (or cardinality) of $E(F_p)$
F_p	the finite field containing exactly p elements
g	a generator of G_q
$gcd(N_1, N_2)$	the greatest common divisor of integers N_1 and N_2
G_q	a cyclic group of prime order q. For uniformity, the multiplicative notation is used throughout. As such, when using the elliptic curve construction, it should be understood that ab represents the group addition of points a and b, that a/b represents the group addition of the point a to the additive inverse of the point b, and that a^b represents the scalar multiplication of point a by the integer b. NOTE This part of ISO/IEC 18370 considers two constructions for the group G_q, in which it is infeasible to compute discrete logarithms. The first is based on a subgroup of a finite field, and the second is based on elliptic curves over a finite field F_q, where q is a prime number. Details of these two constructions are provided in Annex C.
H	a cryptographic hash-function
I	a set of integers
$[n]P$	scalar multiplication operation that takes a positive integer n and a point P on the elliptic curve E as input and produces as output another point Q on the elliptic curve E, where $Q = [n]P = P + P +...+ P$ added n - 1 times. The operation satisfies $[0]P = O_E$ (the point at infinity), and $[-n]P = [n](-P)$.
O_E	the point at infinity on the elliptic curve E
$P + Q$	the elliptic curve sum of points P and Q
q	a prime number satisfying $\|q\| = l_q$ where l_q is a security parameter

Z_p	the set of integers in $[0, p - 1]$ with arithmetic defined modulo p
Z_N^*	the set of integers U with $0 < U < N$ and $gcd(U, N) = 1$, with arithmetic defined modulo N
$\prod_{(i \in I)} a_i$	product of the values a_i for which $i \in I$
$[x, y]$	the set of integers from x to y inclusive, if x, y are integers satisfying $x \leq y$
$\langle \ldots \rangle$	an ordered list of values to be hashed

5 General requirements

In order to use any of the mechanisms specified in this part of ISO/IEC 18370, the following requirements shall be met.

— Each entity involved in a blind signature mechanism shall be aware of the public domain parameters.

— Each entity shall have access to an authentic copy of the necessary public keys, such as the public verification key.

— Each requestor in a traceable blind signature mechanism shall have a distinguishing identifier that is unambiguously bound to the private requestor key. The distinguishing identifier for a requestor can be the public requestor key.

— Both signer and requestor shall have the means to generate integers uniformly at random from a given range. Techniques for generation of sequences of random bits are specified in ISO/IEC 18031. A method for converting a string of bits to an integer in a given range is specified in Annex B.

— A collision-resistant hash-function such as one of those specified in ISO/IEC 10118 shall be used.

Before issuing a blind signature, the signer might wish to authenticate the requestor. ISO/IEC 18370 does not specify mechanisms for entity authentication. For this purpose, the use of one of the mechanisms specified in ISO/IEC 9798 is recommended.

For traceable blind signature mechanisms, this part of ISO/IEC 18370 does not specify in which circumstances a requestor tracing process or a signature tracing process is used.

6 Blind signature mechanisms

6.1 General

Clause 6 specifies a blind signature mechanism.

NOTE The mechanism in Clause 6 is based on Reference [23] and the associated security analysis is given in Reference [26].

6.2 Mechanism 1

6.2.1 Security parameters

The following symbols apply in the specification of this mechanism:

— k, l_q: security parameters.

The parties should agree on the security parameters in use. Guidance for parameter choice is given in Annex E.

6.2.2 Key generation process

The key generation process of a blind signature mechanism consists of the following procedures:

a) generating domain parameters;

b) generating a private signature key and a public verification key.

The first procedure is executed once when the domain is set up. The second procedure is executed for each signer within the domain, where the outputs are a private signature key and the corresponding public verification key.

The set of domain parameters includes the following parameters:

— q: a prime number where $|q| = l_q$;

— G_q: a cyclic group of prime order q;

— g_1: a random generator of G_q;

— g_2: a random generator of G_q different from g_1;

NOTE 1 An example of recommended parameters for typical security levels is provided in E.2.

NOTE 2 A method for selecting random generators is given in ISO/IEC 14888-3:2006, D.2.2.

— H: a hash-function that outputs a k-bit message digest.

The pair of keys of the signer is computed as follows.

a) The signer picks two integers, x_1 and x_2, uniformly at random from the range $[1, q - 1]$.

b) The signer computes $y = g_1{}^{-x_1} g_2{}^{-x_2}$.

The signature key is the pair (x_1, x_2) and the verification key is y.

6.2.3 Blind signature process

A blind signature process is an interactive protocol between a signer and a requestor. By executing the signing protocol, the requestor obtains a valid signature of a message of the requestor's choice in such a way that the signer learns nothing about the message or the resulting signature.

The signature process involves the following steps. The message to be blindly signed is denoted by m, where $m \in \{0, 1\}^*$.

a) The signer picks two integers, w_1 and w_2, uniformly at random from the range $[0, q - 1]$.

b) The signer computes $a = g_1{}^{w_1} g_2{}^{w_2}$.

c) The signer sends a to the requestor.

d) The requestor receives a from the signer.

e) The requestor picks an integer α uniformly at random from the range $[0, q - 1]$.

f) The requestor picks an integer β uniformly at random from the range $[0, q - 1]$.

g) The requestor picks an integer γ uniformly at random from the range $[0, q - 1]$.

h) The requestor computes $a' = a\, g_1{}^{\alpha} g_2{}^{\beta} y^{-\gamma}$.

i) The requestor computes $c' = H(m \,||\, a')$.

j) The requestor computes $c = c' + \gamma \bmod q$.

k) The requestor sends c to the signer.

l) The signer receives c from the requestor.

m) The signer computes $r_1 = w_1 + c\,x_1 \bmod q$.

n) The signer computes $r_2 = w_2 + c\,x_2 \bmod q$.

o) The signer sends r_1 and r_2 to the requestor.

p) The requestor receives r_1 and r_2 from the signer.

q) The requestor checks that the values r_1 and r_2 have been correctly computed by verifying that $a = {g_1}^{r_1} {g_2}^{r_2} y^c$. If this verification fails, the requestor outputs reject and stops.

r) The requestor computes $r_1' = r_1 + \alpha \bmod q$.

s) The requestor computes $r_2' = r_2 + \beta \bmod q$.

t) The requestor sets the signature to $\sigma = (c', r_1', r_2')$.

6.2.4 Verification process

On input of a message, m, a signature $\sigma = (c', r_1', r_2')$, domain parameters, and the verification key, y, the verification process involves the following steps.

a) The verifier computes $a'' = {g_1}^{r_1'} {g_2}^{r_2'} y^{c'}$.

b) The verifier computes $c'' = H(m \,\|\, a'')$.

c) If $c'' = c'$ then return 1 (valid).

d) Else return 0 (invalid).

7 Blind signature mechanisms with partial disclosure

7.1 General

Clause 7 specifies two blind signature mechanisms with partial disclosure.

NOTE The mechanism in 7.2 is based on Reference [14], in which security proofs can also be found. The mechanism given in 7.3 is based on a scheme originally specified in Reference [17] and the associated security analysis is given in Reference [18].

7.2 Mechanism 2

7.2.1 Security parameters

The following symbol applies in the specification of this mechanism:

— l_q: a security parameter.

7.2.2 Key generation process

The key generation process of a blind signature mechanism consists of the following procedures:

a) generating domain parameters;

b) generating a private signature key and a public verification key.

The set of domain parameters includes the following parameters:

— q: a prime number satisfying $|q| = l_q$;

— G_q: a cyclic group of prime order q;

— g: a random generator of G_q;

— F: $\{0, 1\}^* \rightarrow G_q$ a cryptographic hash-function, where the discrete logarithm of value $F(x)$ in base g shall be unknown;

— H: $\{0, 1\}^* \rightarrow [0, q - 1]$ a hash-function.

NOTE 1 An example of recommended parameters for typical security levels is provided in E.2.

NOTE 2 Examples of how to construct F and H are provided in Annex D.

The pair of keys of the signer is computed as follows.

a) The signer picks an integer, x, uniformly at random from the range $[1, q - 1]$.

b) The signer computes $y = g^x$.

The signature key is x and the verification key is y.

7.2.3 Blind signature process with partial disclosure

The signature process involves the following steps. The message to be blindly signed is denoted by m, where $m \in \{0, 1\}^*$, and the common information is denoted by *info*, where $info \in \{0, 1\}^*$.

a) The signer picks three integers, u, s, d, uniformly at random from the range $[0, q - 1]$, and computes $z = F(info)$.

b) The signer computes $a = g^u$, $b = g^s z^d$.

c) The signer sends a, b to the requestor.

d) The requestor receives a, b from the signer.

e) The requestor picks two integers, t_1 and t_2, uniformly at random from the range $[0, q - 1]$.

f) The requestor picks two integers, t_3 and t_4, uniformly at random from the range $[0, q - 1]$.

g) The requestor computes $z = F(info)$.

h) The requestor computes $a' = ag^{t_1} y^{t_2}$, $b' = bg^{t_3} z^{t_4}$.

i) The requestor computes $e' = H(a' \,||\, b' \,||\, z \,||\, m) \in [0, q - 1]$.

j) The requestor computes $e = e' - t_2 - t_4 \bmod q$.

k) The requestor sends e to the signer.

l) The signer receives e from the requestor.

m) The signer computes $c = e - d \bmod q$.

n) The signer computes $r = u - cx \bmod q$.

o) The signer sends r, c, s, d to the requestor.

p) The requestor receives r, c, s, d from the signer.

q) The requestor checks that the values r, c, s, d have been correctly computed by verifying that $a = g^r y^c$, $b = g^s z^d$, $e = c + d \bmod q$. If this verification fails, the requestor outputs reject and stops.

r) The requestor computes $r' = r + t_1$, $c' = c + t_2 \bmod q$.

s) The requestor computes $s' = s + t_3$, $d' = d + t_4 \bmod q$.

t) The requestor sets the signature to $\sigma = (r', c', s', d')$.

7.2.4 Verification process

On input of a message, m, a common information, *info*, a signature, $\sigma = (r', c', s', d')$, domain parameters, and the verification key, y, the verification process involves the following steps.

a) The verifier computes $z = F(info)$, $a' = g^{r'} y^{c'}$, $b' = g^{s'} z^{d'}$.

b) The verifier computes $e' = H(a' \parallel b' \parallel z \parallel m)$.

c) If $e' = c' + d' \bmod q$ then return 1 (valid).

d) Else return 0 (invalid).

7.3 Mechanism 3

7.3.1 Symbols

The following symbol applies in the specification of this mechanism:

— l_q: a security parameter.

7.3.2 Key generation process

The key generation process of a blind signature mechanism consists of the following procedures:

a) generating domain parameters;

b) generating a private signature key and a public verification key.

The set of domain parameters includes the following parameters:

— q: a prime number satisfying $|q| = l_q$;

— G_q: a cyclic group of prime order q;

— g_1: a random generator of G_q;

— g_2: a random generator of G_q different from g_1;

— H: $\{0, 1\}^* \rightarrow [0, q - 1]$ a cryptographic hash-function;

— H_1: $\{0, 1\}^* \rightarrow [0, q - 1]$ a cryptographic hash-function.

NOTE 1 An example of recommended parameters for typical security levels is provided in E.2.

NOTE 2 Examples of how to construct H and H_1 are provided in Annex D.

The pair of keys of the signer is computed as follows.

a) The signer picks an integer, x, uniformly at random from the range $[1, q - 1]$.

b) The signer computes $y_1 = g_1{}^x$.

c) The signer computes $y_2 = g_2{}^x$.

The signature key is x and the verification key is the pair (y_1, y_2).

7.3.3 Blind signature process with partial disclosure

The signature process involves the following steps. The message to be blindly signed is denoted by m, where $m \in \{0, 1\}^*$, and the common information is denoted by *info*, where $info \in \{0, 1\}^*$.

a) The signer picks an integer, ω, uniformly at random from the range $[0, q - 1]$.

b) The signer computes $g_M = g_1^{H_1(info)} g_2$.

c) The signer computes $t' = g_M^{\omega}$.

d) The signer sends t' to the signer.

e) The requestor receives t' from the signer.

f) The requestor picks two integers, λ and μ, uniformly at random from the range $[0, q - 1]$.

g) The requestor computes $g_M = g_1^{H_1(info)} g_2$.

h) The requestor computes $y_M = y_1^{H_1(info)} y_2$.

i) The requestor computes $t_M = t' \, g_M^{\lambda} \, y_M^{\mu}$.

j) The requestor computes $c = H(t_M \,||\, info \,||\, m)$.

k) The requestor computes $c' = c - \mu \bmod q$.

l) The requestor sends c' to the signer.

m) The signer receives c' from the requestor.

n) The signer computes $r' = \omega - c'x \bmod q$.

o) The signer sends r' to the requestor.

p) The requestor receives r' from the signer.

q) The requestor checks that the value r' has been correctly computed by verifying that $t' = g_M^{r'} \, y_M^{c'}$. If this verification fails, the requestor outputs reject and stops.

r) The requestor computes $r = r' + \lambda \bmod q$.

s) The requestor sets the signature to $\sigma = (c, r)$.

7.3.4 Verification process

On input of a message, m, a common information, *info*, a signature, $\sigma = (c, r)$, domain parameters, and the verification key, (y_1, y_2), the verification process involves the following steps.

a) The verifier computes $t'' = \left(g_1^{H_1(info)} g_2\right)^r \left(y_1^{H_1(info)} y_2\right)^c$.

b) The verifier computes $c'' = H(t'' \,||\, info \,||\, m)$.

c) If $c = c''$ then return 1 (valid).

d) Else return 0 (invalid).

8 Blind signature mechanisms with selective disclosure

8.1 General

Clause 8 specifies a blind signature mechanism with selective disclosure.

NOTE The mechanism in Clause 8 is based on Reference [16].

8.2 Mechanism 4

8.2.1 Security parameters

The following symbol applies in the specification of this mechanism:

— l_q: a security parameter.

8.2.2 Key generation process

The key generation process of a blind signature mechanism with selective disclosure consists of the following procedures:

a) generating domain parameters;

b) generating a private signature key and a public verification key.

The first procedure is executed once when the domain is set up. The second procedure is executed for each signer within the domain. The outputs are a private signature key and the corresponding public verification key.

The set of domain parameters includes the following parameters:

— q: a prime number satisfying $|q| = l_q$;

— G_q: a cyclic group of prime order q;

— g: a random generator of G_q;

— n: an integer indicating the number of messages to be signed by the signer;

— $g_1,\ldots, g_n, g_t$: $n + 1$: random generators of G_q, different from each other's and from g, and where t is a special index for the token message.

The pair of keys of the signer is computed as follows.

a) The signer picks an integer, y_0, uniformly at random from the range $[1, q - 1]$.

b) The signer computes $g_0 = g^{y_0}$.

The signature key is the element y_0 and the verification key is g_0.

8.2.3 Blind signature process with selective disclosure

A blind signature process with selective disclosure is an interactive process between a signer and a requestor. By executing the signing process, the requestor obtains a valid signature on a vector of messages known to the signer, and a message and public key known only to the requestor, in such a way that the signer learns nothing about the resulting signature. The resulting public key and signature is called a token. The token can be presented to a verifier using the corresponding private key as described in 8.2.5.

The signature process involves the following steps. The message vector to be signed is denoted by $(x_1,..., x_n, x_t)$ where the $x_i \in [0, q - 1]$ (t is a special token index), and the requestor's message is denoted by PI, where $PI \in \{0, 1\}^*$.

a) The signer and the requestor both compute $\gamma = g_0 g_1{}^{x_1} ... g_n{}^{x_n} g_t{}^{x_t} \in G_q$.

b) The signer computes $\sigma_z = \gamma^{y_0}$.

c) The signer picks an integer, w, uniformly at random from the range $[0, q - 1]$.

d) The signer computes $\sigma_a = g^w$.

e) The signer computes $\sigma_b = \gamma^w$.

f) The requestor picks an integer, α, uniformly at random from the range $[1, q - 1]$.

g) The requestor picks an integer, β_1, uniformly at random from the range $[0, q - 1]$.

h) The requestor picks an integer, β_2, uniformly at random from the range $[0, q - 1]$.

i) The requestor computes $h = \gamma^\alpha$.

j) The requestor computes $t_1 = g_0{}^{\beta_1} g^{\beta_2}$.

k) The requestor computes $t_2 = h^{\beta_2}$.

l) The requestor computes $\alpha^{-1} \bmod q$.

m) The signer sends $(\sigma_z, \sigma_a, \sigma_b)$ to the requestor.

n) The requestor receives $(\sigma_z, \sigma_a, \sigma_b)$ from the signer.

o) The requestor computes $\sigma'_z = \sigma_z{}^\alpha$.

p) The requestor computes $\sigma'_a = t_1\, \sigma_a$.

q) The requestor computes $\sigma'_b = \sigma'_z{}^{\beta_1} t_2 \sigma_b{}^{\alpha}$.

r) The requestor computes $\sigma'_c = H(h \,||\, PI \,||\, \sigma'_z \,||\, \sigma'_a \,||\, \sigma'_b) \bmod q$.

s) The requestor computes $\sigma_c = \sigma'_c + \beta_1 \bmod q$.

t) The requestor sends σ_c to the signer.

u) The signer receives σ_c from the requestor.

v) The signer computes $\sigma_r = \sigma_c\, y_0 + w \bmod q$.

w) The signer sends σ_r to the requestor.

x) The requestor receives σ_r from the signer.

y) The requestor computes $\sigma'_r = \sigma_r + \beta_2 \bmod q$.

z) The requestor verifies that $\sigma'_a \sigma'_b = (gh)^{\sigma'_r} (g_0 \sigma'_z)^{-\sigma'_c}$. If this verification fails, the requestor outputs reject and stops.

aa) The requestor outputs the token consisting of the public key, h, and the signature $(\sigma'_z, \sigma'_c, \sigma'_r)$, and the corresponding private key, α^{-1}.

8.2.4 Presentation process

A blind signature with selective disclosure presentation process is performed by the requestor. By executing the presentation process, the requestor creates a presentation proof containing a subset of the signed message vector.

The presentation process takes the following input parameters:

— $D \subseteq \{1, ..., n\}$ the set of disclosed messages indices;

— $U = \{1, ..., n\} \setminus D$ the set of undisclosed messages indices;

— $(m, m_d) \in \{0, 1\}^*$, the two part message to be signed by the requestor;

NOTE The message to be signed is separated into two parts, m and m_d, to allow extension mechanisms, not defined herein, to use a second-factor to sign part of the protocol message, m_d, without seeing the all of the protocol details.

— h, the token public key;

— $\alpha^{-1} \in [0, q - 1]$, the token private key;

— $(x_1,..., x_n)$, the signed messages where the $x_i \in [0, q - 1]$.

The presentation process involves the following steps.

a) The requestor picks an integer, w_0, uniformly at random from the range $[0, q - 1]$.

b) For each $i \in U$, the requestor picks an integer, w_i, uniformly at random from the range $[0, q - 1]$.

c) The requestor computes $a = H\left[h^{w_0} \prod_{(i \in U)} g_i^{\,w_i}\right]$.

d) The requestor computes $UID_t = H(h \,||\, \sigma'_z \,||\, \sigma'_c \,||\, \sigma'_r)$.

e) The requestor computes $c_p = H(UID_t \,||\, a \,||\, \langle D \rangle \,||\, \langle \{x_i\} i \in D \rangle \,||\, Ø \,||\, Ø \,||\, Ø \,||\, Ø \,||\, Ø \,||\, Ø \,||\, m)$ where Ø is the null value, a zero-length octet string.

f) The requestor computes $c = H\left(\left\langle c_p, m_d \right\rangle\right) \bmod q$.

g) The requestor computes $r_0 = c\,\alpha^{-1} + w_0 \bmod q$.

h) For each $i \in U$, the requestor computes $r_{\mathrm{i}} = -c\, x_i + w_i \bmod q$.

i) The requestor outputs the presentation proof $\{x_i\}_{i \in D}$, x_t, a, r_0, $\{r_i\}_{i \in U}$.

8.2.5 Verification process

A blind signature with selective disclosure verification process is performed by the verifier.

The verification process takes the following input parameters:

— $D \subseteq \{1, ..., n\}$ the set of disclosed messages indices;

— $U = \{1, ..., n\} - D$ the set of undisclosed messages indices;

— $(m, m_d) \in \{0, 1\}^*$, the two part message signed by the requestor;

NOTE The message to be signed is separated into two parts, m and m_d, to allow extension mechanisms, not defined herein, to use a second-factor to sign part of the protocol message, m_d, without seeing all of the protocol details.

— h, the token public key;

— $PI \in \{0, 1\}^*$, the requestor's information message;

— $(\sigma'_z, \sigma'_c, \sigma'_r)$, the token signature;

— $\{x_i\}_{i \in D}, x_t, a, r_0, \{r_i\}_{i \in U}$, the presentation proof.

The verification process involves the following steps.

a) The verifier verifies that $h \neq 1$. If this verification fails, the verifier outputs invalid and stops.

b) The verifier verifies that $\sigma'_c = H\left(h \| PI \| \sigma'_z \| g^{\sigma'_r} g_0{}^{-\sigma'_c} \| h^{\sigma'_r} \sigma'_z{}^{-\sigma'_c}\right)$ mod q. If this verification fails, the verifier outputs invalid and stops.

c) The verifier computes $UID_t = H(h \| \sigma'_z \| \sigma'_c \| \sigma'_r)$.

d) The verifier computes $c_p = H(UID_t \| a \| \langle D \rangle \| \langle \{x_i\} i \in D \rangle \| \emptyset \| \emptyset \| \emptyset \| \emptyset \| \emptyset \| \emptyset \| m)$.

e) The verifier computes $c = H\left(\langle c_p, m_d \rangle\right)$ mod q.

f) The verifier verifies that $a = H\left(\left[g_0 g_t{}^{x_t} \prod_{(i \in D)} g_i{}^{x_i}\right]^{-c} h^{r_0} \left[\prod_{(i \in U)} g_i{}^{r_i}\right]\right)$. If this verification fails, the verifier outputs invalid, and valid otherwise.

9 Traceable blind signature mechanisms

9.1 General

Clause 9 specifies a traceable blind signature mechanism.

NOTE The mechanism in Clause 9 is based on Reference [19], in which security proofs can also be found.

9.2 Mechanism 5

9.2.1 Symbols

The following symbol applies in the specification of this mechanism:

— l_q: a security parameter.

9.2.2 Key generation process

The key generation process of a fair blind signature mechanism consists of five procedures that shall be executed in the following order for mechanism 5:

a) generation of domain parameters;

b) generation of the requestor tracing key and the corresponding public requestor tracing key;

c) generation of the signature tracing key and the corresponding public signature tracing key;

d) generation of a private signature key and a public verification key;

e) generation of the private requestor key and the corresponding public requestor key.

The set of domain parameters includes the following parameters:

— q: a prime number satisfying $|q| = l_q$;

— G_q: a cyclic group of prime order q;

— b: a random generator of G_q;

— g, g_1 and g_2: three random generators of G_q different from each other's and from b;

— H: $\{0, 1\}^* \rightarrow [0, q - 1]$ a cryptographic hash-function.

NOTE 1 An example of recommended parameters, for typical security levels, is provided in E.2.

NOTE 2 Examples of how to construct H are provided in Annex D.

The pair of keys of the requestor tracing authority is computed as follows.

a) The requestor tracing authority picks an integer, x_{RT}, uniformly at random from the range $[1, q - 1]$.

b) The requestor tracing authority computes $y_{RT} = b^{x_{RT}}$.

The requestor tracing key is x_{RT} and the public requestor tracing key is y_{RT}.

The pair of keys of the signature tracing authority is computed as follows.

— The signature tracing authority picks an integer, x_{ST}, uniformly at random from the range $[1, q - 1]$.

— The signature tracing authority computes $y_{ST} = g^{x_{ST}}$.

The signature tracing key is x_{ST} and the public signature tracing key is y_{ST}.

The pair of keys of the signer is computed as follows.

a) The signer picks an integer, x, uniformly at random from the range $[1, q - 1]$.

b) The signer computes $y = y_{ST}^x$.

The signature key is x and the verification key is y.

The pair of keys of the requestor is computed as follows.

— The requestor picks an integer, x_R, uniformly at random from the range $[1, q - 1]$.

— The requestor computes $y_R = g_1^{x_R}$.

The private requestor key is x_R and the public requestor key is y_R. The public requestor key will also be the distinguishing identifier of the requestor.

9.2.3 Traceable blind signature process

The signature process involves the following steps. The message to be blindly signed is denoted by m, where $m \in \{0, 1\}^*$.

a) The requestor sends y_R to the signer.

b) The signer receives y_R from the requestor.

c) The signer computes a challenge, $DT \in [0, q - 1]$. This challenge shall include a nonce, that is, a unique number that is never reused; a large random number will do, as will a timestamp or a counter appended to a unique signer identifier.

d) The requestor receives DT from the signer.

e) The requestor picks five random integers, $s, t, \alpha, \beta, \gamma$, uniformly at random from the range $[0, q - 1]$.

f) The requestor computes $E_1' = y_{RT}^s \, y_{ST}^t$.

g) The requestor computes $E_2 = g^t$.

h) The requestor computes $E_1 = y_R\, E_1'$.

i) The requestor computes $T_1 = g_1^{\alpha}\, y_{RT}^{\,\beta}\, y_{ST}^{\gamma}$.

j) The requestor computes $T_2 = g^{\gamma}$.

k) The requestor computes $c_1 = H(E_1' \parallel E_2 \parallel y_R \parallel DT \parallel T_1 \parallel T_2)$.

l) The requestor computes $r_1 = \alpha - c_1\, x_R \bmod q$.

m) The requestor computes $r_2 = \beta - c_1 s \bmod q$.

n) The requestor computes $r_3 = \gamma - c_1 t \bmod q$. The tuple (c_1, r_1, r_2, r_3) is denoted by $Proof_1 = (c_1, r_1, r_2, r_3)$.

o) The requestor sends E_1', E_2, c_1, r_1, r_2, r_3 to the signer.

p) The signer receives E_1', E_2, c_1, r_1, r_2, r_3 from the requestor. The signer then checks the validity of $Proof_1$ by executing the following steps.

   1) The signer computes $E_1'' = y_R\, E_1'$.

   2) The signer computes $T_1' = g_1^{\,r_1} y_{RT}^{\,r_2} y_{ST}^{\,r_3} E_1''^{\,c_1}$.

   3) The signer computes $T_2' = g^{\,r_3} E_2^{\,c_1}$.

   4) The signer computes $c' = H(E_1' \parallel E_2 \parallel y_R \parallel DT \parallel T_1' \parallel T_2')$.

   5) The signer then checks whether c' is equal to c_1 or not. If this verification fails, the signer outputs invalid and stops.

q) The signer picks an integer, ω, uniformly at random from the range $[1, q - 1]$.

r) The signer computes $m_0 = g_2 E_1''$.

s The signer computes $z_0 = m_0^{\,x}$.

t) The signer computes $A_0 = y_{ST}^{\,\omega}$.

u) The signer computes $B_0 = m_0^{\,\omega}$.

v) The signer sends z_0, A_0 and B_0 to the requestor.

w) The requestor receives z_0, A_0 and B_0 from the requestor.

x) The requestor picks four integers, u, v, η, δ, uniformly at random from the range $[1, q - 1]$.

y) The requestor computes $m_0 = g_2 E_1$.

z) The requestor computes $s_I = y_R\, g_2\, y_{RT}^{\,s} = m_0 \,/\, y_{ST}^{\,t}$.

aa) The requestor computes $A = A_0^{\,u}\, y_{ST}^{\,v}$.

bb) The requestor computes $A_2 = b^s$.

cc) The requestor computes $B = B_0^{\,u}\, m_0^{\,v} \,/\, A^t$.

dd) The requestor computes $D = g_1^{\,\eta}\, y_{RT}^{\,\delta}$.

ee) The requestor computes $E = b^{\delta}$.

ff) The requestor computes $z = z_0 \,/\, y^t$.

gg) The requestor computes $c = H(s_I \parallel z \parallel A \parallel B \parallel A_2 \parallel D \parallel E \parallel m)$.

hh) The requestor computes $c_0 = c \,/\, u \bmod q$.

ii) The requestor sends c_0 to the signer.

jj) The signer receives c_0 from the requestor.

kk) The signer computes $r_0 = \omega - c_0 x \bmod q$.

ll) The signer sends r_0 to the requestor.

mm) The requestor receives r_0 from the signer.

nn) The requestor checks that the value r_0 has been correctly computed by verifying that $A_0 = {y_{ST}}^{r_0} y^{c_0}$ and that $B_0 = {m_0}^{r_0} {z_0}^{c_0}$. If this verification fails, the requestor outputs reject and stops.

oo) The requestor computes $r = u\, r_0 + v \bmod q$.

pp) The requestor computes $r_4 = \eta - cx_R \bmod q$.

qq) The requestor computes $r_5 = \delta - cs \bmod q$. The pair (r_4, r_5) is denoted by $Proof_2 = (r_4, r_5)$.

rr) The requestor sets the signature of the message, m, as $\sigma = (z, c, r, r_4, r_5, s_I, A_2, D, E)$, where s_I corresponds to the signature identifier of this signing session. The output of the signer is the transcript TS of the signing session: $TS = (y_R, DT, E_1', E_2, c_1, r_1, r_2, r_3)$.

9.2.4 Verification process

On input of a message m, a signature $\sigma = (z, c, r, r_4, r_5, s_I, A_2, D, E)$, domain parameters, the public requestor tracing key, the public signature tracing key, and the verification key, the verification process involves the following steps:

a) The verifier computes $A' = {y_{ST}}^r y^c$

b) The verifier computes $B' = {s_I}^r z^c$

c) The verifier computes $c''' = H(s_I \| z \| A' \| B' \| A_2 \| D \| E \| m)$. The signer then checks whether c''' is equal to c or not. If this verification fails, the signer outputs invalid and stops

d) The verifier then checks whether $E = b^{r_5} {A_2}^c$ and $D = {g_1}^{r_4} {y_{RT}}^{r_5} (s_I / g_2)^c$ hold. If this verification fails, the signer outputs invalid and valid otherwise.

9.2.5 Requestor tracing process

On input of a valid traceable blind signature, $\sigma = (z, c, r, r_4, r_5, s_I, A_2, D, E)$, domain parameters, the requestor tracing key and the verification key, the requestor tracing process involves the following steps.

a) The requestor tracing authority computes $G = {A_2}^{x_{RT}}$.

b) The requestor tracing authority computes the distinguishing identifier, d_{ID}, of the requestor who requested the traceable blind signature, $\sigma = (z, c, r, r_4, r_5, s_I, A_2, D, E)$: $d_{ID} = (s_I / g_2) / G = y_R$.

c) **[Optional]** The requestor tracing authority creates the requestor tracing evidence EB, which demonstrates that the distinguishing identifier, d_{ID}, has been correctly computed.

   1) The requestor tracing authority picks an integer, τ, uniformly at random from the range $[0, q - 1]$.

   2) The requestor tracing authority computes $Z = {A_2}^{\tau}$.

   3) The requestor tracing authority computes $Z' = b^{\tau}$.

   The requestor tracing authority computes $\varepsilon = H(z \| c \| r \| r_4 \| r_5 \| s_I \| A_2 \| D \| E \| Z \| Z')$.

4) The requestor tracing authority computes $\rho = \tau - \varepsilon\, x_{RT} \bmod q$ and outputs the requestor tracing evidence *EB*: $EB = (\varepsilon, \rho)$.

d) The requestor tracing authority outputs the distinguishing identifier, d_{ID}, and optionally, if required, the requestor tracing evidence, *EB*.

9.2.6 Signature tracing process

On input of the transcript of a signing session $TS = (y_R, DT, E_1', E_2, c_1, r_1, r_2, r_3)$, domain parameters, the signature tracing key and the public key of the signer, the signature tracing process involves the following steps.

a) The signature tracing authority computes $F = E_2^{x_{ST}}$.

b) The signature tracing authority computes the signature identifier, s_I, which will allow to identify the signature yielded from the signing session whose transcript is *TS*: $s_I = g_2 y_R (E_1'/F)$.

c) [**Optional**] The signature tracing authority creates signature tracing evidence, *LP*, which demonstrates that the signature identifier, s_I, has been correctly computed.

1) The signature tracing authority picks an integer, μ, uniformly at random from the range $[0, q - 1]$.

2) The signature tracing authority computes $W = E_2^{\mu}$.

3) The signature tracing authority computes $W' = g^{\mu}$.

4) The signature tracing authority computes $\pi = H(y_R \,||\, DT \,||\, E_1' \,||\, E_2 \,||\, c_1 \,||\, r_1 \,||\, r_2 \,||\, r_3 \,||\, W \,||\, W')$.

5) The signature tracing authority computes $\phi = \mu - \pi\, x_{ST} \bmod q$ and outputs the signature tracing evidence *LP*: $LP = (\pi, \phi)$.

d) The signature tracing authority outputs the signature identifier s_I and, if required, the signature tracing evidence *LP*.

9.2.7 Requestor tracing evidence evaluation process

On input of requestor tracing evidence $EB = (\varepsilon, \rho)$, a valid traceable blind signature $\sigma = (z, c, r, r_4, r_5, s_I, A_2, D, E)$, domain parameters, and the public requestor tracing key, the requestor tracing evidence evaluation process involves the following steps.

a) The requestor tracing evidence evaluator computes $Z'' = A_2^{\rho}\, G^{\varepsilon}$.

b) The requestor tracing evidence evaluator computes $Z''' = b^{\rho}\, y_{RT}^{\varepsilon}$.

c) The requestor tracing evidence evaluator computes $\varepsilon' = H(z \,||\, c \,||\, r \,||\, r_4 \,||\, r_5 \,||\, s_I \,||\, A_2 \,||\, D \,||\, E \,||\, Z'' \,||\, Z''')$.

d) If $\varepsilon = \varepsilon'$ then return 1 (valid).

e) Else return 0 (invalid).

9.2.8 Signature tracing evidence evaluation process

On input of signature tracing evidence $LP = (\pi, \phi)$, the transcript of a signing session $TS = (y_R, DT, E_1', E_2, c_1, r_1, r_2, r_3)$, domain parameters, and the public signature tracing key, the signature tracing evidence evaluation process involves the following steps:

a) The signature tracing evidence evaluator computes $W'' = E_2^{\phi}\, F^{\pi}$.

b) The signature tracing evidence evaluator computes $W''' = g^{\phi}\, y_{ST}^{\pi}$.

c) The signature tracing evidence evaluator computes $\pi' = H(y_R \,||\, DT \,||\, E_1' \,||\, E_2 \,||\, c_1 \,||\, r_1 \,||\, r_2 \,||\, r_3 \,||\, W'' \,||\, W''')$.

d) If $\pi = \pi'$ then return 1 (valid).

e) Else return 0 (invalid).

Annex A
(normative)

Object identifiers

Annex A lists the object identifiers assigned to the blind digital signature mechanisms specified in this part of ISO/IEC 18370.

```
BlindDigitalSignaturesMechanisms-2 {
iso(1) standard(0) blind-digital-signatures-mechanisms(18370) part2(2)
asn1-module(0) object-identifiers(0) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN
-- EXPORTS All; --
-- IMPORTS None; --
OID ::= OBJECT IDENTIFIER -- alias
-- Synonyms --
is18370-2 OID ::= { iso(1) standard(0) blind-digital-signatures-mechanisms
(18370) part2(2) }
mechanism OID ::= { is18370-2 mechanisms(2) }
-- blind digital signatures mechanisms --
bs-discrete-logarithm-representation-proof OID ::= { mechanism 1 }
bs-partial-disclosure-or-proof OID ::= { mechanism 2 }
bs-partial-disclosure-equality-proof OID ::= { mechanism 3 }
bs-selective-disclosure-discrete-logarithm-proof OID ::= { mechanism 4 }
bs-traceable-verifiable-encryption-ElGamal OID ::= { mechanism 5 }
END -- BlindDigitalSignaturesMechanisms-2 –
```

Annex B
(normative)

Conversion functions

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

— The function BS2IP(x) maps a bit string, x, to an integer value, m, as follows. If $x = (x_{l-1}, \ldots, x_0)$ where $x_0, \ldots, xl-1$ are bits, then the value m is defined as $m = 2^{l-1}x_{l-1} + 2^{l-2}\,x_{l-2} + \ldots + 2x_1 + x_0$.

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

Annex C
(normative)

Group description

This part of ISO/IEC 18370 defines two constructions for the group G_q. Either constructions may be used for the mechanisms described in this part of ISO/IEC 18370. Each construction is specified by a description desc(G_q):

— **Subgroup construction**: The description desc(G_q) = (p, q, g) specifies a subgroup G_q of prime order q of a finite field of order, p. Both p and q are prime numbers, q divides $p - 1$, and g is a generator of G_q. It is recommended to use the method defined in ISO/IEC 14888-3:2006, Annex D to generate the group description (p, q, g).

— **Elliptic curve construction**: The description desc(G_q) = (p, a, b, g, q, 1) specifies an elliptic curve over a finite field F_p, where p is a prime number, a and b are two field elements defining the elliptic curve, g is a base point of prime-order q on the curve (and the generator of G_q), q is the order of the group, and 1 is the cofactor of the curve, which implies that $\#E(F_p) = q$. Methods of generating pseudo-random elliptic curves and points of prime order q (the order of the elliptic curve E) are given in ISO/IEC 15946-5 and examples of pseudo-random elliptic curves are given in ISO/IEC 15946-5:2009, C.1.

All entities involved in the mechanisms described in this part of ISO/IEC 18370 should check that all externally received mathematical elements belong to their corresponding algebraic structures prior to relying on or computing with them; failure to do so may result in critical security or privacy problems as pointed out, for example, in Reference [20] or Reference [25]. For an element $x \in Z_q$, this means verifying that $0 \leq x < q$. For an element $x \in G_q$, it is sufficient to make sure the curve equation holds when using the elliptic curve construction and to verify that $0 < x < p$ and that $x^q = 1$ when using the subgroup construction.

NOTE Only curves with prime order are allowed for the elliptic curve construction. Since the cofactor is 1 for curves of prime order, all curve points are part of the group, and therefore checking that the curve equation holds is enough to verify that a point is part of the group.

To mitigate attacks of the type described in Reference [20] or Reference [25], p and q should be selected, in the subgroup construction, so that $(p - 1)/(2 \times q)$ has no prime factor less than q. Ideally, $(p - 1) / (2 \times q)$ should be prime.

Annex D
(informative)

Special hash-functions

D.1 Hash-function with larger output length: HL

HL is a cryptographic function that hashes a string, m, into $\{0, 1\}^k$ based on a hash-function $H: \{0, 1\}^* \rightarrow \{0, 1\}^h$ in ISO/IEC 10118, where $k > h$. HL is constructed using MGF1 in PKCS#1. It involves the following steps.

a) If $k > 2^{32}h$, output “Fail” and stop.

b) Let T be an empty binary string.

c) For i from 0 to $\lceil k / h \rceil - 1$, set $T = T \,||\, H(m \,||\, \text{I2BSP}(i, 32))$.

d) Return the leading k bits of T.

D.2 Hashing to an element of a prime field: HBS2PF

HBS2PF is a cryptographic function that hashes a string m into an element in Z_p.

HBS2PF2 involves the following steps.

a) Let H be a hash-function in ISO/IEC 10118 that outputs at least the same bit length as p.

b) Let $h = \text{BS2IP}(H(m))$.

c) Return h mod p.

D.3 Hashing to a point on an elliptic curve: HBS2ECP

Let E be an elliptic curve over an explicitly given prime field, F_p. HBS2ECP is a cryptographic function that hashes a string, m, into a point in E. It involves the following steps.

a) Let $i = 0$.

b) Let I2ECP be a primitive that converts integers to elliptic curve points in ISO/IEC 15946-1.

c) Let $x = \text{HBS2PF}(\text{I2BSP}(i, 32) \,||\, m)$.

d) Let $P = \text{I2ECP}(x)$. If I2ECP succeeds, output P and quit the procedure.

e) Increment i by 1. If $i < 2^{32}$, then go to step c), otherwise return “Fail”.

D.4 Hashing to an element of a cyclic group: HBS2CG

HBS2CG is a cryptographic function that hashes a string, m, into an element in a cyclic group G_q. Two constructions of such hash-function are given in this Annex: the first one works with cyclic groups G_q based on a subgroup of a finite field (subgroup construction), whereas the second one works for cyclic group G_q based on elliptic curves over a prime field (elliptic curve construction). They are denoted as $\text{HBS2CG}_{\text{SG}}$ and $\text{HBS2CG}_{\text{EC}}$, respectively.

Subgroup construction: $HBS2CG_{SG}$ involves the following steps.

a) Let p and q be two primes, such that q divides $p - 1$ but q^2 does not divide $p - 1$.

b) Let HL be a hash-function, as defined in D.1, that outputs at least the same bit length as p.

c) Let $r = (p - 1) / q$.

d) Let $i = 0$.

e) Let $h = \mathrm{BS2IP}(\mathrm{HL}(\mathrm{I2BSP}(i, 32) \parallel m))^r \bmod p$. If $h > 1$ output h and quit the procedure.

f) Increment i by 1. If $i < 2^{32}$, then go to step e), otherwise return “Fail”.

Elliptic curve construction: $HBS2CG_{EC}$ involves the following steps.

a) Let E be an elliptic curve over an explicitly given prime field F_p as defined in Annex C (elliptic curve construction).

b) Let $i = 0$.

c) Let $h = \mathrm{HBS2ECP}(m)$. If HBS2ECP succeeds and $h \neq O_E$, output h and quit the procedure.

d) Increment i by 1. If $i < 2^{32}$, then go to step c), otherwise return “Fail”.

Annex E
(informative)

Security considerations and comparison of blind signature mechanisms

E.1 Descriptions of mathematical assumptions

E.1.1 General

The following computational hardness assumptions underlie the security (one-more unforgeability and blindness) of the mechanisms specified in this part of ISO/IEC 18370; namely, the discrete logarithm (DL) assumption[23] and the decisional Diffie-Hellman (DDH) assumption.[15]

NOTE The blindness property, namely the unlinkability of the signatures issued by the signer, hold unconditionally for mechanisms 1, 2, 3 and 4, limited only by the quality of the requestor-generated random numbers. It relies on the DDH assumption for mechanism 5.

E.1.2 The discrete logarithm (DL) assumption

The DL assumption is the assumption that the following problem is hard to solve. Given a cyclic group G_q of order q; g, a generator of G_q and $y \in G_q$, find x such that $y = g^x$.

E.1.3 The decisional Diffie-Hellman (DDH) assumption

The DDH assumption is the assumption that the following problem is hard to solve. Given a cyclic group G_q of order q and three elements, g^a, g^b, $z \in G_q$, decide whether $z = g^{ab}$.

E.2 Guidance for parameters choice

E.2.1 Key sizes

The performance and the security and privacy properties of the mechanisms described in this part of ISO/IEC 18370 depend on the sizes of elements in G_q and in Z_q. In particular, the discrete logarithm problem (or the DDH problem for mechanism 5) should be hard in G_q.

When using the subgroup construction, minimum sizes for p and q of 2048 bits and 256 bits, respectively, are recommended for long-term security. The level of security of the mechanisms described in this part of ISO/IEC 18370 that results from a particular choice for the sizes of p and q is believed to be the same as that for DSS (see Reference [21]).

When using the elliptic curve construction, the level of security is believed to be the same as in ECDSA (also see Reference [21]). For general guidelines on picking key sizes to ensure the infeasibility of computing discrete logarithms or solving the DDH problem in G_q, see Reference [22], section 5.6.

E.2.2 Hash algorithm selection and digest sizes

The hash algorithm specified in the domain parameters of mechanism 1 should be a cryptographically secure hash algorithm, meaning that it should be collision-resistant and behave as much as possible as a “random oracle”.

Furthermore, the digest size of this hash algorithm (which is equal to k for mechanism 1) should be close or equal to the size of q. A collision-resistant hash-function, matching the size of q, such as one of those specified in ISO/IEC 10118, should be used. In light of state-of-the-art attacks on hash-functions, a 256-bit q is recommended.

E.2.3 Random number generation

The strength of the security and privacy provided by the mechanisms described in this part of ISO/IEC 18370 critically depend on both the quality and secrecy of the random numbers used by each entity involved in these mechanisms. A robust random bit generator such as one of those specified in ISO/IEC 18031 should therefore be used.

E.3 Symbols for comparing each mechanism

For the purposes of Annex E, the following symbols apply (subgroup construction).

α	bit-length of a prime number (or prime power) q
β	bit-length of a prime number (or prime power) p
exp	modular exponentiation (mod q)
m_q	modular multiplication (mod q)
m'_q	modular inversion (mod q)
m_p	modular multiplication (mod p)
m'_p	modular inversion (mod p)
n	number of messages (mechanism 4)

E.4 Comparison of each mechanism

Table E.1 — Efficiency of each mechanism

		Mechanism 1	Mechanism 2	Mechanism 3	Mechanism 4	Mechanism 5
Nature of blind signature		Blind signature	Blind signature with partial disclosure	Blind signature with partial disclosure	Blind signature with selective disclosure	Traceable blind signature
Key size	Signature key	2α bits	α bits	α bits	α bits	α bits
	Verification key	β bits	β bits	2β bits	β bits	β bits
Signature size		3α bits	4α bits	2α bits	$(2\alpha + \beta)$ bits	$(4\alpha + 5\beta)$ bits
Computation	Key generation	2 exp, 1 m_p	1 exp	2 exp	1 exp	4 exp (all keys)
	Signature generation	8 exp, 2 m_q, 6 m_p	11 exp, 1 m_q, 7 m_p	8 exp, 1 m_q, 6 m_p	$(2n + 12)$ exp, 1 m_q, 1 m'_q, $(2n + 10)$ m_p	31 exp, 7 m_q, 1 m'_q, 18 m_p, 2 m'_p
	Signature verification	3 exp, 2 m_p	4 exp, 2 m_p	4 exp, 3 m_p	$(n + 7)$ exp, $(n + 4)$ m_p	9 exp, 5 m_p, 1 m'_p

Annex F
(informative)

Numerical examples

Annex F provides numerical examples for each blind signature mechanism specified in this part of ISO/IEC 18370.

F.1 Mechanism 1

F.1.1 Generation of domain parameters

This example makes use of a cyclic group G_q that is a subset of Z_p*. The hash-function used is SHA-256.

```
k   = 256 (decimal)

lq  = 256 (decimal)

p   = ca863be0 e5ba677a aa728cb9 67128abb 5e27d82a aec80778 9d3058ac d58b0d0d

      f38715e3 7829893e a8df495c a49d8f96 8bb668ee 72a62482 5be22372
      7eb07949

      29bf0eec 33212014 8dbee767 54a41ab0 465adde1 d9bc592f 6d8cec13
      52da5af3

      bc6ddf25 e6898bcf 9ef65c3b 2f3bd373 8bb6fdfd 7b5e367d a4df7067
      330bf9e1

      7c374d13 749c9ff3 98a3a675 1a29b589 5d9d064a d96a86d3 810cc687 8a6b2b3c

      4b56302b 221e31ca 12bb2116 d8a5fc5a abeb143b 4ef7219b e221076f
      802ced8c

      cc7def9e 2de9d3fe 7fd34969 a406a753 3bcb326b c0913e85 1e4700ba
      2403fb65

      f206f5f1 b20b4eaa 83cfb034 77ae57ba 88901cdc 1a523768 b7f2e133
      b2e6068a

      ac446c48 4f96bf42 b57ba354 556f6b8e 5ccbc746 f09bd34c e23983b8
      d77ee84e

      f1f2cc82 e153da85 a81b3597 10fe6828 78c848e9 1cf73e0e 98261e96
      423d61a1

      f3f7ddf0 931b459e a6c5e354 f3dd435b ab8d87f7 50e52c17 26123104
      a65a47e2

      523033c9 bae45ce6 b531a450 5c9fb813 11918ec8 047c285f e57ba60e
      9bb92997

q   = 8f40a65d 5449388b 3d1da48a 150d5f43 ef7e401c 27d75a2e 57bb666c
      3b9f0e9b

g1  = 5a9e3f83 5ebebac5 ab17959a c806c807 59160c2a 7bda079b 269d5278
      4387a1af
```

```
          d753452d 0196a7d7 f20577be d6745289 c5d21d48 66182fa5 19870e14
          f677ee2e

          c77bd08a 8c8549af 369f3236 86fa2068 d3e0f195 adc515f5 e0486952
          657d9678

          160f7eda 39e6c1f3 4f47c202 0d0ff9e0 b7e79e70 30535ed9 5f871d56
          025df294

          3d6dfb93 7fc36154 a74cc4ce c3d5ec08 6df6d706 b01d0326 a3a258d1
          d1e28014

          09fe70cf c383a64d 20b0e348 a3a3877c 0a5711a8 8fe6072b 6f7aab8a a5be28fc

          ecb05f89 1c11ac65 1b826c1d 00fadd4b 53340de4 6cd530bb 2e50acb4
          d99b099c

          40089588 d50a467c 82c503ff 1396d540 17188b18 55683828 b9f84942 f9adcb08

          ada2c488 ab1ab9ed c71c71c0 25ec1f52 5b6a3123 dc86a548 083fa13b
          2984a717

          d00f111e 57a3d657 aead2eaf 1490eb3f 9a860625 7a5c8c3c 8b107e70
          29a2be10

          241b932a 0754edf8 5ed22b18 093bee10 3ebe0022 25ef90ee eff8bd16
          c4c2e83f

          0ddf24f9 cde59ccb 87a5a706 5c9acc98 61b2e0b1 65d05887 ca636622 e39ab3cb

g2      = 341ab3de 3245b472 55c5e4e8 64bfcc27 59dd87d3 53a8a23a e901ec3d
          8171538b

          51ed6ad1 da55d1fd d9d82ca5 49eb162b b13a5464 6e6a2dce f1dfaf64
          60f574e1

          2fcad8f6 8d4d1064 145fe5d9 1b7b278c 92aa5e6c 26e75e14 282b0b52
          e1cbbbf0

          81e3aa45 c9b88479 ba252908 b892baf4 69088431 1afa6e85 f76d2316
          09f916bc

          181646f1 f0055b28 22371519 d260f478 d3f29ce5 12fbab7f d5301bad
          028a5d54

          b56103a5 44c9726d f00ce268 fd882c48 cfcb0e35 31b22879 e2cf3b1a
          f53964bd

          8c6bb43a ba5ef40a 9036e454 e0c23c2c 7bc82448 6ecb3ad8 10ca348f
          4d6021fe

          d15ec7b2 6862b659 009600ad 90fed7e8 226c5ed4 34f083fb ef39b028
          1f7235a7

          aaf2c875 9cd1e961 ea4a9bca 23abafda 99b81c10 03d46d1a e680ed01
          8511df02
```

```
        cd26ac73 3e400315 05e12b23 8bed34ae 095c6366 a336c439 8830c7e5
        23bf01fd

        0083a796 8df1d10f f0805e38 0df72147 f5290b4a 835929f9 d511efae
        06b62ba4

        cfb59170 e2c8a34a fd8ef792 fdcb5873 d5766b40 5ffd9406 d08b41ff
        f514f207
```

F.1.2 Generation of signature key and verification key

```
x1  = 19929b08 7163d13b a0dd0040 37014c21 909ba773 8e211ead 5fc6f076
      85d4b4ff

x2  = 33a1d327 2f28f5bd ce3cff6d b88a72a7 437c0fe6 cbbffa73 8db23dd4
      f6c71eac

y   = 1219db74 cad167cb 3bf9c312 561cfa8d afe916f6 5aa2c9bf e2855407
      c23d1741

      509e6e53 66f3172f d22ed5ee a5b63a98 6926328c 2ef6d9e1 711839e3
      96b061d3

      0d78af89 f6315b91 d1a5d45d 7dbcff00 f0dc5678 42009166 38d6673a
      64dcc308

      77964784 9e31028d 8b5a7e2c fceebb0a d07b7e12 04afc55c bd8f2048
      2d75d7ca

      f23bb2ec 5a166a0e 22c20a68 47d36c29 e4111b3b e6b5eafa 140e46e3
      1775a7bb

      9160f208 5b03f333 a3aa713d 30bdf36f b2decdc3 74865a69 549ae4a4
      803928a5

      e8d3dfbd 7481dbf5 b7bdc6ec c2a5326e f22bb20a d1dbb47c 9b04527d
      49b18e14

      8fb8d6af 6db4ef2b b48afc87 bbfbd0b3 9841eb34 39447730 23ee38d6
      b6a4f4e9

      8cdaf997 c56157d9 8e819e3a ba12fdcb b0cc8cd7 3c997bf6 1d654086
      225d330e

      508ded17 d6b55845 ee00a9fa 07c1a5d3 b9d160e0 8c09542d 2a26b175
      10c08ce9

      816967f9 a8c2c903 ea970996 6a187528 9f7054ce e2f479ec 48a63778
      03835685

      46256172 aac187d4 d0d6122f 5ebab3c5 e22f67ea b8e0e6f5 7408336e
      7d34e467
```

F.1.3 Blind signature process

Message: “This is the message unknown to the signer who will produce a blind signature through an interaction with the requestor.”

```
m      = 54686973 20697320 74686520 6d657373 61676520 756e6b6e 6f776e20
         746f2074

         68652073 69676e65 72207768 6f207769 6c6c2070 726f6475 63652061
         20626c69

         6e642064 69676974 616c2073 69676e61 74757265 20746872 6f756768
         20616e20

         696e7465 72616374 696f6e20 77697468 20697473 20726563
         69706965 6e742e

m.len  = 1016 (decimal)

w1     = 4878475c cbfe889b b55f069c 31407747 433fa00b cf16a60c 2e5f8987
         0f47db22

w2     = 5fdfc242 178bd938 817c6975 8c7a929d 7d14880c f0334b0a eef03ef8
         3a88d0f8

a      = 0eba2027 e0318873 4c5a3094 69e187d4 a8e9b157 c9ba3f23 e3fd9036
         ea2a45d0

         01c8f354 f080daed 310e8760 faf418f7 a3ca0d54 2abadc1b 1616565c
         6e87a4aa

         e8214970 dcad1976 142e6664 b05df0ce e4ffc5d8 dd92214a 11fd4c27
         3a770c9f

         1163142a 4690a957 0da7d59a 9fa5bd84 6ae630ed b3df5a58 0e5f21f1
         ba5d6652

         2e96468e 8b835341 d032ed09 91be13bc 617115cc c2478c62 d83ed01d
         7a2abd44

         188b607f e4c04315 fe64dfb2 21780a8d 483d9c49 4a573e2f 55d0f906
         bb3799a5

         5dad2f07 92dc7c42 4825ced7 f45d6bc4 5820d345 0131c9c8 bad22559
         32757bbe

         edb73738 19a1ba90 56ecbe45 23cbfdd0 d60e1532 25d8aeeb 04a7db03
         f804f45a

         400f7e40 334f6d16 d7bb1d88 2c140bbd fe98db97 ece20fff 488ef0af
         0407ce54

         309bce44 c51060ff 618d874f fb10f373 63fff318 94aa6be6 0f83d94d
         49d7066d

         4b831c73 3b0da296 18f900a1 b20287c4 62adef82 8c5a7ca7 7ca483f5
         78e03986

         4113ea3f a1bd6a6f 913cc952 dcd4e001 d74acab2 7e68fdf8 d39f7c02
         6bae37c3

alpha  = 6c04e84f 3acb271b d6380fee 37f40911 7e8c4b64 73f3a445 4422e489
         4cadf4b9

beta   = 7a8a9864 6c9b7aad 26ffdb42 e30feb65 1f149dd4 19a14e4e 52ea6b08
         7de3f2f9
```

```
gamma  = 2630d5b1 78e35baa 9f67d19f 5c8e98e6 9bc62535 a6b56a7f 3df6836b
         9d69f1ca

a'     = 84d0c1f9 48892a51 c8c08a36 35654492 84745ca4 b670f459 5692afa6
         b0ea09f2

         fa3cfba9 e71385be 5dc7db1a 1a79c7d2 5f203d38 ae5a7a5f 1489f8f5
         3ee23ee6

         0a552a92 1c17e90a 6aeaba70 6484fedc a5305445 344d6749 0b641838
         31345566

         42feebd8 f8e9f483 ebed5904 52752aa2 9f58b48d f17ec2e3 d5b72e21
         d37b95f8

         417f6066 d09695ed 3a65edb1 a9703299 717af1cb c013638c 04bd04a9
         cc165a2a

         395fb706 e9226d74 2d7cd046 1e55f0a1 5d26165f fb422215 d7221fd9
         d42bca02

         0f408305 56131799 2b1ab5a7 e83f5cf6 b0af01b4 28ff4c89 baf35ebc
         89746ed4

         f7a47925 89a35d18 b5f792cb 50df2360 f0a54dcd 76a57a0d 899e0fad
         add32647

         32d264f1 e0e758d0 88b8ca7c 00b6c00e d53f7273 4462a777 8a7585b5
         c86c27c2

         cd142c9a 6a646af4 7116c1ef 239aaa1e 5dc5de28 5385cdda 29994ba2
         c1eb13b8

         08861eca 06aff8f9 6e6004ca d830472e 014ebf96 c0893323 54d65448
         8c614965

         c44f32fa 5c63fa73 252365e8 8b037a3c 91bebd8c a1f505df b62b6062
         925957c2

m||a'  = 54686973 20697320 74686520 6d657373 61676520 756e6b6e 6f776e20
         746f2074

         68652073 69676e65 72207768 6f207769 6c6c2070 726f6475 63652061
         20626c69

         6e642064 69676974 616c2073 69676e61 74757265 20746872 6f756768
         20616e20

         696e7465 72616374 696f6e20 77697468 20697473 20726563 69706965
         6e742e84

         d0c1f948 892a51c8 c08a3635 65449284 745ca4b6 70f45956 92afa6b0
         ea09f2fa

         3cfba9e7 1385be5d c7db1a1a 79c7d25f 203d38ae 5a7a5f14 89f8f53e
         e23ee60a

         552a921c 17e90a6a eaba7064 84fedca5 30544534 4d67490b 64183831
         34556642

         feebd8f8 e9f483eb ed590452 752aa29f 58b48df1 7ec2e3d5 b72e21d3
         7b95f841
```

```
           7f6066d0 9695ed3a 65edb1a9 70329971 7af1cbc0 13638c04 bd04a9cc
           165a2a39

           5fb706e9 226d742d 7cd0461e 55f0a15d 26165ffb 422215d7 221fd9d4
           2bca020f

           40830556 1317992b 1ab5a7e8 3f5cf6b0 af01b428 ff4c89ba f35ebc89
           746ed4f7

           a4792589 a35d18b5 f792cb50 df2360f0 a54dcd76 a57a0d89 9e0fadad
           d3264732

           d264f1e0 e758d088 b8ca7c00 b6c00ed5 3f727344 62a7778a 7585b5c8
           6c27c2cd

           142c9a6a 646af471 16c1ef23 9aaa1e5d c5de2853 85cdda29 994ba2c1
           eb13b808

           861eca06 aff8f96e 6004cad8 30472e01 4ebf96c0 89332354 d654488c
           614965c4

           4f32fa5c 63fa7325 2365e88b 037a3c91 bebd8ca1 f505dfb6
           2b606292 5957c2

c’       = 3c8d0ebf b9265bfc 7c61b3d3 8882a8ae 178af97a d9e0ab9b 262d2011
           03ee7b98

c        = 62bde471 3209b7a7 1bc98572 e5114194 b3511eb0 8096161a 6423a37c
           a1586d62

r1       = 2173d29a e27948a9 9c4e3d1b 489a421d 22fab90d bc56ac95 1320fe86
           34cf05e8

r2       = 82d2797f be66f7e5 41365884 abe5fe3c fcc341ef 3a2ed3a7 380361d2
           e66bc68a

Values of r1 and r2 are accepted.

r1’      = 8d78baea 1d446fc5 72864d09 808e4b2e a1870472 304a50da 5743e30f
           817cfaa1

r2’      = 6e1c6b86 d6b93a07 2b188f3d 79e88a5e 2c599fa7 2bf8c7c7 3332666f
           28b0aae8
```

F.1.4 Verification process

```
a’’ = 84d0c1f9 48892a51 c8c08a36 35654492 84745ca4 b670f459 5692afa6
      b0ea09f2

      fa3cfba9 e71385be 5dc7db1a 1a79c7d2 5f203d38 ae5a7a5f 1489f8f5
      3ee23ee6

      0a552a92 1c17e90a 6aeaba70 6484fedc a5305445 344d6749 0b641838
      31345566

      42feebd8 f8e9f483 ebed5904 52752aa2 9f58b48d f17ec2e3 d5b72e21
      d37b95f8

      417f6066 d09695ed 3a65edb1 a9703299 717af1cb c013638c 04bd04a9
      cc165a2a

      395fb706 e9226d74 2d7cd046 1e55f0a1 5d26165f fb422215 d7221fd9
      d42bca02
```

```
        0f408305 56131799 2b1ab5a7 e83f5cf6 b0af01b4 28ff4c89 baf35ebc
        89746ed4

        f7a47925 89a35d18 b5f792cb 50df2360 f0a54dcd 76a57a0d 899e0fad
        add32647

        32d264f1 e0e758d0 88b8ca7c 00b6c00e d53f7273 4462a777 8a7585b5
        c86c27c2

        cd142c9a 6a646af4 7116c1ef 239aaa1e 5dc5de28 5385cdda 29994ba2
        c1eb13b8

        08861eca 06aff8f9 6e6004ca d830472e 014ebf96 c0893323 54d65448
        8c614965

        c44f32fa 5c63fa73 252365e8 8b037a3c 91bebd8c a1f505df b62b6062
        925957c2

c’’   = 3c8d0ebf b9265bfc 7c61b3d3 8882a8ae 178af97a d9e0ab9b 262d2011
        03ee7b98

Verification successful.
```

F.2 Mechanism 2

F.2.1 Finite field based domain parameters

```
p length:2048

q length:224

Key generation
```

F.2.1.1 Generation of domain parameters

```
hash function: SHA-256

p     = EA8E526D A73D95E6 B7BFF5CE 62DADB0E 77EF2011 18E05B7B 804B3D90
        9D41FA39

        D5DF3CE7 36B2AEA5 8480543C 6D187193 426D4996 15191D1A E3CE1949
        0299F473

        3C4B93DB 85E91C61 353E4EB3 A483DADB 28268856 70A175C5 3A792E6E
        FB351A80

        11F08FF4 7754E153 37EB8B3F 928D5E89 0CE25758 1E26C963 A8CE38A8
        91BBA962

        BB1F80C4 37C212C2 ADD6E1F3 294EE66D E35188B3 7EF7C183 0EB9CDFE
        470A6094

        9D1191F4 04D6E14C B11B8DB9 5D43F7F3 354FE7CA FBBFE6E5 DF648FEB
        711CB761

        09FDE8DC 17CA229C F2F318AC 2A2E318F 9A9FA539 08DCFAC3 D6990A1C
        8E5A91D7

        BAFCCD3A 94492E0B DE9E7FCB F9BC9EA6 EFBEF0B5 6D8BFE7D 2F1B447E
        07675CF9

q     = D3FD9625 302930D4 93F92545 BD863E5C A5AB44E1 252F231A 7AF22059
```

F.2.1.2 Generation of signature key and verification key

```
x    = 6305E3A9 17009CBD 9C870360 804EEB0D 2A2A3DE5 13CA8220 0C197AE6

y    = 0CB3A43F DBBC6CA0 682A0C4F E85A1F81 B31D2426 0C41BAD5 892C787B
       AFC496FB

       65939C23 2F54FF2E 1697329E DD787BF6 81121D72 7704F7BC F629FFC3
       E0C2D630

       B6624D05 600BC978 B5F8611D 69BFFB40 35BC2E6B 45BD8C7B DEFE8228
       7E2BACC5

       CA2DF009 3BFA5F80 9391E37F E14FFE1F E4A0BD15 C1A0AE14 DE3F4782
       D9632AA8

       D0A794F0 6175935D 98697E83 DE37628F 5E67C506 101AF916 6B724FAD
       D2164857

       0D0C7CCE 42E5A033 B7AEA691 90958397 92CD7774 937AD185 D5564594
       2D4FA4DB

       34961613 FDA28770 686F49A4 923B0663 0A1BD1CC DE095943 7B20DFB8
       029764A2

       3C53D86B 1C72AB02 643E81F5 EAAF3155 6AF024DB 873F6794 BF352754
       23E6694A
```

F.2.1.3 Blind signature process with partial disclosure

```
Message = This is the message unknown to the signer who will produce a
blind
signature through an Interaction with the requestor.

Common info = This is the common information.

m      = 54686973 20697320 74686520 6D657373 61676520 756E6B6E 6F776E20
         746F2074

         68652073 69676E65 72207768 6F207769 6C6C2070 726F6475 63652061
         20626C69

         6E642064 69676974 616C2073 69676E61 74757265 20746872 6F756768
         20616E20

         496E7465 72616374 696F6E20 77697468 20746865 20726563 69706965
         6E742E

m.len  = 1016 bits

info   = 54686973 20697320 74686520 636F6D6D 6F6E2069 6E666F72 6D617469
         6F6E2E
```

Signer:

```
u      = 7784105D 6A7A4003 8CDCA46C C12BC24F AB297B19 3019EC8D EA3811C6

s      = 9DD7BCC6 A39FD5EA E5D80D26 E6AB2A07 D9AD9649 8BB1970F D326F865

d      = 27960864 FAC273EB 44BF89A1 50CA01F2 EF28C3AF AE6314C6 92B054DF

z      = 227E666C 7724086E BC3BD4D7 5B33D3C1 9BD98207 F97C33E2 05CE8FD6
         DE17DF2C
```

```
          C52656A0 42CEC1C0 1CD1CB7C F23CB282 9EBCE35A 0D0FB18B 54474022
          134EF3FB

          FDD45052 021CFCE4 2C1F9A02 6E27B763 494CAA94 EC670038 B122BA27
          11C293C2

          918E376D 6DE245F4 27EF526D FF25BD7F 56AE22BE 04A1A875 F54AAC2B
          B81DFE4D

          76B6243A C747F14E 2D8D309B D972FA17 F5B21471 A358AB20 5435D546
          4569738C

          2FAE4342 ED0DB189 C15CB3C3 186246E6 92B88B29 310022F0 636FC54B
          B2BC5D5B

          5709460A D5C7BFE7 6D99976A C427DCBC F9862DC4 15800A45 8CCA62F5
          185E358D

          2AF182D4 F1A2BF2E 0B9A439F B25D83DA 5A1653A6 EB4433F9 91EFA492
          45DD2F42

a       = 9E86D38F 9471533E 983C173B 3F1334AF 539617BB D1470EBE DFA85293
          C69994EE

          807CD411 0CB50039 D994914E 19148304 D1419468 C73EA983 3F4BD176
          8AB46D9C

          CD08E71C FEDC77C1 AC1A896B 24A25E2E 8CD8C3C9 9C638F62 94C7F9AB
          18578C2B

          A00EECCB A08A63C9 1DB8EE1F BAF9B7F2 D6223BC6 D8DB05B9 002BF20C
          A4B44D4B

          BF515AF4 2BAD6364 3DA7585A 5906CCFD 8E33F0C2 BBDE100C 26C64F10
          AD93CD5B

          48A48673 0D39D569 ABCC3C59 A7C9FA57 0F1EDEAF 0A4C6EE2 A6A57C45
          D772790D

          38F32BB2 4C71FFD9 39608C06 2A4C13CB D4625E06 CB00B3C0 79FF46D2
          3ADCF23A

          6FAA46FC 3BD41CA7 0F94B327 6AD3F172 A9ADB7BC C599EA18 33E7D4E1
          3D763295

b       = D3FF6297 FDA78F68 E67222DE B2E3D236 8B04A341 7ABF72E3 50925FF9
          170CD61E

          75321A71 48C30463 1F092E8F E49571D0 84B47F93 9CC50FBE 332AE723
          73ECBEBD

          3B3F3C18 31B874BD 66D5AEA0 74896AE8 21297C86 294AF041 6A7DA2E3
          977CFAF2

          32A18E30 92A1E0A0 4D238330 EBEB3DE1 EE9668FD 64B5578D C9D03044
          3AB7DDC6

          A92023FF 13DEC72F CE0A4556 ABCE4FE2 A552DE8E 385C0C41 50067D32
          F960421C

          2A09F85C 1530A5DC 70CC6FEA 7AEFDCDB 1B137D9F D5672BBB 859829D2
          DFC10C52
```

```
         57A9D08C BE6262EE 12BE320D 04C5F1D1 FF20B637 76F4F401 E5CF17FE
         E1117ED0

         4997859C 7345E330 D26B01C7 9C603A3E FF96BD5C 23DAB87B 27E93246
         39CDD01E
```

Requestor:

```
t1     = 7E5318C9 EBAE8502 33C9723A 913C68F0 63D7D3B5 E2817C9C 1B31EC75

t2     = 1197F87E 1F8E15AD D1A1F96D 9CD5C061 609E0594 318724FC 140D890B

t3     = CE659A94 3D293A3A E8F95050 115C2E1E 43A01F47 A450E85F E55CC295

t4     = 35B0A1D9 EE209B76 FD242319 0C681234 AA770D07 E5C96402 103C72D1

z      = 227E666C 7724086E BC3BD4D7 5B33D3C1 9BD98207 F97C33E2 05CE8FD6
         DE17DF2C

         C52656A0 42CEC1C0 1CD1CB7C F23CB282 9EBCE35A 0D0FB18B 54474022
         134EF3FB

         FDD45052 021CFCE4 2C1F9A02 6E27B763 494CAA94 EC670038 B122BA27
         11C293C2

         918E376D 6DE245F4 27EF526D FF25BD7F 56AE22BE 04A1A875 F54AAC2B
         B81DFE4D

         76B6243A C747F14E 2D8D309B D972FA17 F5B21471 A358AB20 5435D546
         4569738C

         2FAE4342 ED0DB189 C15CB3C3 186246E6 92B88B29 310022F0 636FC54B
         B2BC5D5B

         5709460A D5C7BFE7 6D99976A C427DCBC F9862DC4 15800A45 8CCA52F5
         185E358D

         2AF182D4 F1A2BF2E 0B9A439F B25D83DA 5A1653A6 EB4433F9 91EFA492
         45DD2F42

a’     = D128E740 8E377438 BC4F5E4D 3AF9FE08 89B795BC 74DAF5E1 92328650
         6BA0FBD2

         14A79BC9 B71771FA 13FE1588 18E043D8 25424694 F0AFC2C6 E8D8DDBD
         DF3CA1BD

         AAC3A728 9A1DD4F7 351AAEFB F009211C 13FFAD77 95C21DA4 16CD54A8
         0A33B5D3

         3DDF490F 17445ABB 255A6FE7 95E16028 8A755A0F E02CDCD5 CC0E970B
         E1DFDB67

         65F43DFE FC3D9465 17048A14 7B522E90 357CAECC AD9C3C42 9CA33B27
         FFD20FEA

         0911DF11 413D853B F2465074 112DD5C4 815950A9 C80CA854 2350E737
         C4DBF8F0

         761FB07A 753AA15C 0490A003 260D2093 55320C4F A5EEC02A 9DABE988
         F6419034

         C86FEDB4 B0307CD1 4B995F5E 1ADE8BD0 10C12454 F9D1DCA3 A0A7A022
         2BE5C805
```

```
b′     = 4E9D1C0A 72E5436F 08074091 42059807 AB6FDECA B5EC3099 3D8D850E
         EFABEAA5

         18426CF4 717D9607 42CA9451 A671E788 50BB5CC3 B407DD8A 3EBFA673
         20CA93E6

         863E5711 9AB44DCB 29BD0E9A DBDE6B61 3A813097 80367AC9 251B72B9
         EF5013C3

         40194862 21FA897A 2AC8D580 CBF12122 96258586 84B972BC A2577A94
         98CE9535

         EDA810D0 C64ED830 9783DE64 9A9294CC 682970AD 36B5C289 B5D2B914
         4BBFAB7E

         447C77B4 7CBEDE34 F743390C ED2FEB2F 4CF86517 D14DFAC0 E7E4E6FB
         6F6771B9

         AB9609A6 DEAC874D 97FB84CD 5BDE2AA0 4B50F472 0D02D141 C9214F3F
         45C488BB

         F62CE173 450BDCCA 64CFC226 0BA5B56B AD743964 9447282A 4EF6D853
         B51866D5

e′     = 3CF19E37 E21842DC B3DD37E4 EB8AE058 A2838850 CC51F5FB 7E17913F

e      = C9A69A05 0492C28C 791040A3 FFD34C1F 3D19BA95 DA309017 D4BFB5BC
```

Signer:

```
c      = A21091A0 09D04EA1 3450B702 AF094A2C 4DF0F6E6 2BCD7B51 420F60DD

r      = 05F7D449 7012D86C FBE8D419 D2EEC639 46A06317 3A31C706 63DFC38A
```

Requestor:

```
a      = 9E86D38F 9471533E 983C173B 3F1334AF 539617BB D1470EBE DFA85293
         C69994EE

         807CD411 0CB50039 D994914E 19148304 D1419468 C73EA983 3F4BD176
         8AB46D9C

         CD08E71C FEDC77C1 AC1A896B 24A25E2E 8CD8C3C9 9C638F62 94C7F9AB
         18578C2B

         A00EECCB A08A63C9 1DB8EE1F BAF9B7F2 D6223BC6 D8DB05B9 002BF20C
         A4B44D4B

         BF515AF4 2BAD6364 3DA7585A 5906CCFD 8E33F0C2 BBDE100C 26C64F10
         AD93CD5B

         48A48673 0D39D569 ABCC3C59 A7C9FA57 0F1EDEAF 0A4C6EE2 A6A57C45
         D772790D

         38F32BB2 4C71FFD9 39608C06 2A4C13CB D4625E06 CB00B3C0 79FF46D2
         3ADCF23A

         6FAA46FC 3BD41CA7 0F94B327 6AD3F172 A9ADB7BC C599EA18 33E7D4E1
         3D763295

b      = D3FF6297 FDA78F68 E67222DE B2E3D236 8B04A341 7ABF72E3 50925FF9
         170CD61E
```

```
           75321A71 48C30463 1F092E8F E49571D0 84B47F93 9CC50FBE 332AE723
           73ECBEBD

           3B3F3C18 31B874BD 66D5AEA0 74896AE8 21297C86 294AF041 6A7DA2E3
           977CFAF2

           32A18E30 92A1E0A0 4D238330 EBEB3DE1 EE9668FD 64B5578D C9D03044
           3AB7DDC6

           A92023FF 13DEC72F CE0A4556 ABCE4FE2 A552DE8E 385C0C41 50067D32
           F960421C

           2A09F85C 1530A5DC 70CC6FEA 7AEFDCDB 1B137D9F D5672BBB 859829D2
           DFC10C52

           57A9D08C BE6262EE 12BE320D 04C5F1D1 FF20B637 76F4F401 E5CF17FE
           E1117ED0

           4997859C 7345E330 D26B01C7 9C603A3E FF96BD5C 23DAB87B 27E93246
           39CDD01E

r'       = 844AED13 5BC15D6F 2FB24654 642B2F29 AA7836CD 1CB343A2 7F11AFFF

c'       = B3A88A1E 295E644F 05F2B070 4BDF0A8D AE8EFC7A 5D54A04D 561CE9E8

s'       = 983FC135 B09FDF51 3AD83831 3A8119C9 77A270B0 0AD35C55 3D919AA1

d'       = 5D46AA3E E8E30F62 41E3ACBA 5D321427 999FD0B7 942C78C8 A2ECC7B0
```

F.2.1.4 Verification process

```
z          = 227E666C 7724086E BC3BD4D7 5B33D3C1 9BD98207 F97C33E2 05CE8FD6
             DE17DF2C

             C52656A0 42CEC1C0 1CD1CB7C F23CB282 9EBCE35A 0D0FB18B 54474022
             134EF3FB

             FDD45052 021CFCE4 2C1F9A02 6E27B763 494CAA94 EC670038 B122BA27
             11C293C2

             918E376D 6DE245F4 27EF526D FF25BD7F 56AE22BE 04A1A875 F54AAC2B
             B81DFE4D

             76B6243A C747F14E 2D8D309B D972FA17 F5B21471 A358AB20 5435D546
             4569738C

             2FAE4342 ED0DB189 C15CB3C3 186246E6 92B88B29 310022F0 636FC54B
             B2BC5D5B

             5709460A D5C7BFE7 6D99976A C427DCBC F9862DC4 15800A45 8CCA52F5
             185E358D

             2AF182D4 F1A2BF2E 0B9A439F B25D83DA 5A1653A6 EB4433F9 91EFA492
             45DD2F42

a'         = D128E740 8E377438 BC4F5E4D 3AF9FE08 89B795BC 74DAF5E1 92328650
             6BA0FBD2

             14A79BC9 B71771FA 13FE1588 18E043D8 25424694 F0AFC2C6 E8D8DDBD
             DF3CA1BD

             AAC3A728 9A1DD4F7 351AAEFB F009211C 13FFAD77 95C21DA4 16CD54A8
             0A33B5D3
```

```
              3DDF490F 17445ABB 255A6FE7 95E16028 8A755A0F E02CDCD5 CC0E970B
              E1DFDB67

              65F43DFE FC3D9465 17048A14 7B522E90 357CAECC AD9C3C42 9CA33B27
              FFD20FEA

              0911DF11 413D853B F2465074 112DD5C4 815950A9 C80CA854 2350E737
              C4DBF8F0

              761FB07A 753AA15C 0490A003 260D2093 55320C4F A5EEC02A 9DABE988
              F6419034

              C86FEDB4 B0307CD1 4B995F5E 1ADE8BD0 10C12454 F9D1DCA3 A0A7A022
              2BE5C805

b'          = 4E9D1C0A 72E5436F 08074091 42059807 AB6FDECA B5EC3099 3D8D850E
              EFABEAA5

              18426CF4 717D9607 42CA9451 A671E788 50BB5CC3 B407DD8A 3EBFA673
              20CA93E6

              863E5711 9AB44DCB 29BD0E9A DBDE6B61 3A813097 80367AC9 251B72B9
              EF5013C3

              40194862 21FA897A 2AC8D580 CBF12122 96258586 84B972BC A2577A94
              98CE9535

              EDA810D0 C64ED830 9783DE64 9A9294CC 682970AD 36B5C289 B5D2B914
              4BBFAB7E

              447C77B4 7CBEDE34 F743390C ED2FEB2F 4CF86517 D14DFAC0 E7E4E6FB
              6F6771B9

              AB9609A6 DEAC874D 97FB84CD 5BDE2AA0 4B50F472 0D02D141 C9214F3F
              45C488BB

              F62CE173 450BDCCA 64CFC226 0BA5B56B AD743964 9447282A 4EF6D853
              B51866D5

e'          = 3CF19E37 E21842DC B3DD37E4 EB8AE058 A2838850 CC51F5FB 7E17913F

e test      = 3CF19E37 E21842DC B3DD37E4 EB8AE058 A2838850 CC51F5FB 7E17913F

test signature: true
```

F.2.2 Elliptic curve based domain parameters

F.2.2.1 Generation of domain parameters

```
hash function: SHA-256

curve name: secp256r1

curve p = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF
          FFFFFFFF

curve a = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF
          FFFFFFFC

curve b = 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E
          27D2604B
```

```
curve q = FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2
          FC632551

g       = (x = 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0
          F4A13945 D898C296,

          y = 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE
          CBB64068 37BF51F5)
```

F.2.2.2 Generation of signature key and verification key

```
x   = A99C3102 45DFA9AC 5E1C4B2F 6FEFA11D 58C2B248 391FE53D 6EE011F3
      DDA1FF32

y   = (x = 352BBD0D 565DBB41 6689AEDC 64DDE406 1943879E 4594649C DEFE16EC
      D930B78D,

      y = C189839B DC3E10B1 AAC7EFE6 CFCF1202 A04D8EAC E7AD06F0 C018B101
      39398B52)
```

F.2.2.3 Blind signature process with partial disclosure

```
Message = This is the message unknown to the signer who will produce a blind
signature through an Interaction with the requestor.
Common info = This is the common information.
m      =  54686973 20697320 74686520 6D657373 61676520 756E6B6E 6F776E20 746F2074
          68652073 69676E65 72207768 6F207769 6C6C2070 726F6475 63652061 20626C69
          6E642064 69676E61 74757265 20746872 6F756768 20616E20 496E7465 72616374
          496E7465 72616374 696F6E20 77697468 20746865 20726563 69706965 6E742E
m.len  =  1016 bits
info   =  54686973 20697320 74686520 636F6D6D 6F6E2069 6E666F72 6D617469 6F6E2E
```

Signer:

```
u      =  83DBE8ED AE9D8DEA C24EB248 0A89E706 A7223EC8 59E8E40C 79208DB4 C02AFEF8
s      =  3A811DA7 9040F87E 2ABDF1C1 969981B3 F1A0A0B8 B12CFACC 2B6E796C FE45312C
d      =  32D1A5DE F0918127 3EDAA550 214F56AB A7D39E13 9FEC012C 676C26DA 37D22FA1
z      =  (x = 51183088 74C07373 DC539246 06FEB91D 805D7428 9C8BD35A 1187A362
          371028B0,
          y = BF7D0441 110A5CBE E8DA23CE FE54CED7 BA99CA9B AB6DDBB0 6FD877F2
          43F90F0A)
a      =  (x = 742C17B2 56E898A2 70B88E0C 19EBBC79 C2ED9F3B F89E0429 45A49733
          99796A87,
          y = 73B5C64A 3C711264 53F5A769 6069800D 3E48A849 B779207A 94DD0A5B
          99A57A33)
b      =  (x = 412B6BE3 C36F814F 29A70E75 DAAC9C6E 88E04E7D 8ED5ED07 AB500F43
          699631FD,
          y = 361335D4 407D02E6 88F9817F 89CE05B2 D425AAC3 7DC6C3E8 71F4AD7D
          EEF6FB17)
```

Requestor:

```
t1     =  1652135C D1EF5F7F 250DF1AD A34D8A36 27EEFDD4 11EB734F DBA373CC 72CADA04
t2     =  271ADCD1 836E6C71 FC37266A A0FE69A6 782245F6 7E960C61 851C6B94 1951EF4E
t3     =  54E0D6EF 6050B557 A94FACFB 5A5E439A 115B2EF8 1A80CB0A 776D54EE A8D567AE
t4     =  EACB73B0 D326D43C BA108F0B 08EDF7CD AF7679AD 163AADF8 488349CD DA604EAC
z      =  (x = 51183088 74C07373 DC539246 06FEB91D 805D7428 9C8BD35A 1187A362
          371028B0,
          y = BF7D0441 110A5CBE E8DA23CE FE54CED7 BA99CA9B AB6DDBB0 6FD877F2
          43F90F0A)
a'     =  (x = 3C0F8BED 61706F35 7B8F999B EFF3FF61 AD6C2564 CD8B12AB 36992A65
          0A1EDAF8,
          y = 60C84DA1 E6BE7613 E7CBCE55 B77AA34B 1C78C7EA 1C0CD827 95057784
          0B9461E9)
b'     =  (x = B5CB3148 724B1343 F5B0BCAB DC59E657 2F51EAE1 DB250A22 95ACE9FA
          CA17E6AE,
          y = 9CD65224 29295015 F12C2728 604785A8 F142E1B8 9F834BAD 4FC34F60
          28760905)
e'     =  D1C477DD 0E56FC9B B40F3C5D B7EFBB07 E60CCD6E 1F922906 3E4EDB86 5CC631F5
e      =  BFDE2759 B7C1BBED FDC786E8 0E035993 7B5B0878 31D90D31 6468F0E7 6577194C
```

Signer:

```
c      =  8D0C817A C7303AC6 BEECE197 ECB402E7 D3876A64 91ED0C04 FCFCCA0D 2DA4E9AB
r      =  21D1C1EE 3883CCB4 AC3B2650 C8A6EA9A B7063C3F E3463223 35429D6D AD9F5DA4
```

Requestor:

```
a      =  (x = 742C17B2 56E898A2 70B88E0C 19EBBC79 C2ED9F3B F89E0429 45A49733
          99796A87,
          y = 73B5C64A 3C711264 53F5A769 6069800D 3E48A849 B779207A 94DD0A5B
          99A57A33)
b      =  (x = 412B6BE3 C36F814F 29A70E75 DAAC9C6E 88E04E7D 8ED5ED07 AB500F43
          699631FD,
          y = 361335D4 407D02E6 88F9817F 89CE05B2 D425AAC3 7DC6C3E8 71F4AD7D
          EEF6FB17)
r'     =  3823D54B 0A732C33 D14917FE 6BF474D0 DEF53A13 F531A573 10E6113A 206A37A8
c'     =  B4275E4C 4A9EA738 BB240802 8DB26C8E 4BA9B05B 10831866 821935A1 46F6D8F9
s'     =  8F61F496 F091ADD5 D40D9EBC F0F7C54E 02FBCFB0 CBADC5D6 A2DBCE5B A71A98DA
d'     =  1D9D1990 C3B85562 F8EB345B 2A3D4E79 9A631D13 0F0F109F BC35A5E5 15CF58FC
```

F.2.2.4 Verification process

```
z      = (x = 51183088 74C07373 DC539246 06FEB91D 805D7428 9C8BD35A
         1187A362 371028B0,

         y = BF7D0441 110A5CBE E8DA23CE FE54CED7 BA99CA9B AB6DDBB0
         6FD877F2 43F90F0A)

a'     = (x = 3C0F8BED 61706F35 7B8F999B EFF3FF61 AD6C2564 CD8B12AB
         36992A65 0A1EDAF8,

         y = 60C84DA1 E6BE7613 E7CBCE55 B77AA34B 1C78C7EA 1C0CD827
         95057784 0B9461E9)

b'     = (x = B5CB3148 724B1343 F5B0BCAB DC59E657 2F51EAE1 DB250A22 95ACE-
         9FA CA17E6AE,

         y = 9CD65224 29295015 F12C2728 604785A8 F142E1B8 9F834BAD
         4FC34F60 28760905)

e'     = D1C477DD 0E56FC9B B40F3C5D B7EFBB07 E60CCD6E 1F922906 3E4EDB86
         5CC631F5

e test = D1C477DD 0E56FC9B B40F3C5D B7EFBB07 E60CCD6E 1F922906 3E4EDB86
         5CC631F5

test signature: true
```

F.3 Mechanism 3

F.3.1 Finite field based domain parameters

F.3.1.1 Generation of domain parameters

```
hash function: SHA-256
p   = EA8E526D A73D95E6 B7BFF5CE 62DADB0E 77EF2011 18E05B7B 804B3D90 9D41FA39
      D5DF3CE7 36B2AEA5 8480543C 6D187193 426D4996 15191D1A E3CE1949 0299F473
      3C4B93DB 85E91C61 353E4EB3 A483DADB 28268856 70A175C5 3A792E6E FB351A80
      11F08FF4 7754E153 37EB8B3F 928D5E89 0CE25758 1E26C963 A8CE38A8 91BBA962
      BB1F80C4 37C212C2 ADD6E1F3 294EE66D E35188B3 7EF7C183 0EB9CDFE 470A6094
      9D1191F4 04D6E14C B11B8DB9 5D43F7F3 354FE7CA FBBFE6E5 DF648FEB 711CB761
      09FDE8DC 17CA229C F2F318AC 2A2E318F 9A9FA539 08DCFAC3 D6990A1C 8E5A91D7
      BAFCCD3A 94492E0B DE9E7FCB F9BC9EA6 EFBEF0B5 6D8BFE7D 2F1B447E 07675CF9
q   = D3FD9625 302930D4 93F92545 BD863E5C A5AB44E1 252F231A 7AF22059
g1  = D1D4E25C B37C340C B403A8D7 40551F3A 3E5F5AA7 FACEFE92 38EBEAD1 9BADC588
      8103484F C4FE888D F87A7176 8F14459D 58E03019 117E79BC 55E5F2BF 1C328037
      BE4D67D9 E037E3D1 31DE497A E13BE1E3 409595B4 3A0546C7 EE0E2A80 7B4F7414
      33EFFA9C E29F2386 E8E816B9 5D9ABECE 83EF7B8A 1804B8BD 822CA004 E49057B2
      C7EB4759 A6889A6A 8174019F 46B44516 D549F2D5 B244D514 52166198 52528890
      11068593 B5811276 41F79132 30853A6A 582BCE74 84EB82CC 206CBB2B 500CA789
      0CF51147 F4C50109 8A338873 0B3F8354 37981249 68AFC0B5 4BD1AAC8 9F4AF8B4
      B483B8C5 27EA85CF C206EC55 0414727B 9A16325E 1CE638FA 9FB78418 AC1D4DA0
g2  = C7D36FBD C58A8056 B973E4D1 7A258D8B EE5A5199 8728CE34 D84E60D6 B3A7C857
      D63B5E49 4F44E881 F550FCD5 9C4FC375 EFDDBD98 72145A64 A22CB331 56A7C816
      78DBA070 3D85FE02 694D8E6C FFBC44CC A3EBCBFF 3363F3C0 3F0C4B4E 22778B82
      652E1C69 425BD1FD 2B13BACA A4576618 4D907885 B230B1B9 69787ED2 D4C00C8A
      492ACFE6 6A1D4E01 DF02AAA0 EABDAE84 A4C6A65A 9844F58C CD8E3E6D 4C26AE93
      A0BD313C 9DAB7F77 14F09D90 9EB0B6C7 12487727 FEB1406A 50182D5C D75FD572
      C7FC1BC6 8E181685 0200DFBE 03E7A719 F69CB39F 46B8339F 57D18AEE 1A33AE81
      3BEDABE0 3C068889 CD267932 9CEE743D 635CD5FD 3AA456E1 80D918A7 EB23E157
```

F.3.1.2 Generation of signature key and verification key

```
x   = 92D2B29C 5C153E2B 1C04BD80 84FC12B7 F2F7A4AF 9D268C6A 366C97BB

y1  = DA2CBCF4 CC7666A0 DAF49673 1CE3255C 335F0193 20D1DE76 34EDC1DB
      3D4575EA

      DEB55A18 3A0F2E44 67075861 25DBD084 E10D3D94 4E727A0D 864C6BC2
      72C4DAEC

      EEFCB832 3BEE9A3D 53288789 FF04C908 5A5BA371 88874931 9AA8820D
      E7193953

      1E9DD981 4396D784 1D73B848 CDC8CBD0 B1D4970C DEADC639 0474FB3F
      7AB63AC4

      ACB06BC3 E03B4A24 CB750075 7819AEAD 1C11F946 4BFAA746 48E5171F
      E1E5684E
      5D1F3CAD 1AC01FBF 3F32E340 4BD69AAF BF12369D EE1D7671 C798F50B
      23FE4277

      A4D76278 D0C0BA82 7A77A819 5B37923B 0323C146 AAB9A15A F2A76292
      F37B058D

      DCE75D29 3902C460 CDB26038 AB620AC3 2255ADD8 99588781 DEDE5DA6
      A7825841

y2  = E1F0E4AF 7C2C45AB 47919DE2 302B9893 9FD0B74F 7A4DBB1B 75B380D2
      7A69AC1F

      CDD0BA2E EF3C19DA CA316C44 51E5CBA0 BE0C32F9 C24F2ED1 2E429B22
      00689D52

      4A23795E EA1C9409 BAFB117B A7B3C2D9 23B3F99C 3A4B0A93 D998AB0F
      6C41CA4D

      2EEBFDA9 F1024FD3 847222A9 7B25835C B0FE268A 015A75D9 DF8AE24E
      FA3B407C

      83ADE3BF 39C6D2C6 48E3F68A B7586C93 ABB49242 C77675C1 65F521C4
      69B06F59

      2821D3C3 FFC3D4CE B2B43081 92C3F618 189215D9 F9394AA2 5CC79330
      27057860

      64835847 02E759B7 AFE97010 2D4765D5 312CDA10 DF1033CF C6D9EECB
      C0522E96

      67CE5C8B 67B1BBA6 73855B06 828105EA 2F240FEF ADC7BCB6 03DE4738
      31B5D571
```

F.3.1.3 Blind signature process with partial disclosure

```
m    = 4d657373616765206e6f742073686f776e20746f207369676e6572

info = 5075626c696320706172742066206d657373616765

ω    = 274A1F95 91351588 471F9F54 4005B6F8 64F8AEA3 4808EB73 9FC60B72

gM   = 7BEA0F04 E9D4F6E8 303F6032 E6386687 7CAB228F A8878C9C 8113CE37
       4FAA6985

       AD3A778E F18BA220 AA8E86EF BF83BF8C EDC11F5A D8DE40A8 0BB138A9
       8A2D9224

       C4DC990C F68B0E08 01B81009 5AA16199 A5735194 E0EEFF96 1BF1379E
       0D0D428B

       BE248700 323A3B8A 2897BF5D 6CCC4081 C8C1A60B 4BD23565 2B246056
       87881894

       8E2E643B F50EE831 485F4925 C3C22E18 ADC9254F 8401B015 B6B8FF0A
       323CB999

       44A616D7 AF1B0297 F8B9A7AA DAA2FE94 0BA7073C 2581EF17 B53F5B16
       11DABCDE

       678E32C4 4663E2EA 0CF451D0 742F85B7 42799B8B 635510FB 1FA613B4
       2A41DA07
```

```
            1B72DB70 12A24D91 0F5FFCA9 ECF3353F 49D70977 386142DD 6444B1B8
            0D025ED3

t'       =  C9ED2888 FA735AE3 F3192B42 C14D3FCA 71179EDA 569D0603 30D7EBD8
            81F63BBD

            1EE6C596 2895525C 32CAA66A 47CDF701 548F3CB0 4BC11D92 BD27D964
            3654AFDF

            CEA6FEFB BE04465F 0EA32E86 61CB90F4 60F8756B 8E0E32F0 C5EF0F74
            6E0160B6

            F9F4DB6D 5CDA21FC 617CF0D3 872D6F5E 7C2DD2D3 B73C04B1 2E783AF2
            6CC3A0C5

            CD5286DD 2A2EE884 37546EC4 7DAF7835 CD895897 D2018ECA F01D72F0
            E01E9521

            D4860053 D39092ED DAF86262 5F6DA22D A626C072 46F4A3F0 AFEA13A4
            B0A5018C

            D2AD47C1 FE16E514 8B571A8A 9E26909B 40ADC8BD 25F0E412 C36ED0B3
            31E05A3C

            D273E6C4 FEC08241 42E469D2 164547A9 61CEBE01 3D4E71CE 33CFE030
            BD32B135

λ        =  A899E974 6E73ECA0 9612C3C3 0CBA005C 48B3CF62 2671139F 0F6FF5D3

µ        =  6D33E231 10226049 EC9E3809 6B274264 79748D03 BE7D0F37 902F1754

gM       =  7BEA0F04 E9D4F6E8 303F6032 E6386687 7CAB228F A8878C9C 8113CE37
            4FAA6985

            AD3A778F F18BA220 AA8E86EF BF83BF8C EDC11F5A D8DE40A8 0BB138A9
            8A2D9224

            C4DC990C F68B0E08 01B81009 5AA16199 A5735194 E0EEFF96 1BF1379E
            0D0D428B

            BE248700 323A3B8A 2897BF5D 6CCC4081 C8C1A60B 4BD23565 2B246056
            87881894

            8E2E643B F50EE831 485F4925 C3C22E18 ADC9254F 8401B015 B6B8FF0A
            323CB999

            44A616D7 AF1B0297 F8B9A7AA DAA2FE94 0BA7073C 2581EF17 B53F5B16
            11DABCDE

            678E32C4 4663E2EA 0CF451D0 742F85B7 42799B8B 635510FB 1FA613B4
            2A41DA07

            1B72DB70 12A24D91 0F5FFCA9 ECF3353F 49D70977 386142DD 6444B1B8
            0D025ED3

yM       =  2625C572 9413F67E 6ABBF3A7 06D3C0F8 C2C9C2E8 F0AC5CAA A791DAD9
            F22F9345

            C91B9385 135AF690 B70F4A5C 32632236 0497A51A 3F7FD8A3 D1857FA8
            3908B39D

            D94B2CC0 C3403662 BB88BE07 393694F3 3892B4F8 8C0B8818 E9CF9CAC
            66702E73
```

```
           B902A60C 4EFBD583 14058B2E C2536D33 33BE89CD 74AE8484 9D64ECFF
           2EE4751E

           A09C636B 254BB97E CD94911A E30538D6 29649073 9EA4AE92 BDFFF915
           041C39D7

           DB1B060C C4771B7E F6B9FFAF D20DB7FC 2CE01769 661BFA88 E0C441CD
           1794D8DB

           B4E8D8C9 950529B5 90102CF4 91720FC4 BE9D0C36 A73E9615 985D7BF1
           D1D3A5DB

           50485D9B 2717483F CDF7609D 82F6E961 F117039E 81CCC940 78F1FE29
           BA1F717F

tM       = 834763DE CD777788 F3112E05 30A21688 38D774D0 16D277F1 D6707D38
           57E3B652

           0F8C1BDE C2C29760 B4CE4566 D7D1A537 9AB5FF16 1643CB15 43DEC151
           D16AC5D1

           1516349B 5877F8AB 3AB768C7 62E82DFC 1E4FD282 B1735A9D 5958CA56
           A75EFEE0

           2FC0B8BA 2ED1B6D5 E5257021 C6AF6EAD 3AD57A4D 0209992C 7BB1F8E9
           35488563

           EC26FB95 94391FCE 715E9227 AAE8999C 2F6746E8 98722229 424ED0CF
           F10ECFA0

           EB42935B 580662CD FE4CC762 9CEE161B 0B018350 0629AF16 0D483659
           14460E62

           F9CA2238 B9D74EB0 E5172699 E50FF917 D69BE42F 121167D0 D79B4B75
           63CB39B4

           C5D3F35D DD1FE475 3A08F620 A7F37601 2589E335 2FD1FB63 0A600677
           20715311

c        = 3251BBB8 815EBBD3 3A63DABE 3549A407 BB4E7CBA 95A4A6E3 DF9E2BAB

c’       = 991B6FAC A1658C5D E1BEC7FA 87A89FFF E7853497 FC56BAC6 CA6134B0

r’       = 5707E0CD 2B06AD5B 40BFA372 C6414717 147BB43D EB80A7DB 3B9159D5

t’       = C9ED2888 FA735AE3 F3192B42 C14D3FCA 71179EDA 569D0603 30D7EBD8
           81F63BBD

           1EE6C596 2895525C 32CAA66A 47CDF701 548F3CB0 4BC11D92 BD27D964
           3654AFDF

           CEA6FEFB BE04465F 0EA32E86 61CB90F4 60F8756B 8E0E32F0 C5EF0F74
           6E0160B6

           F9F4DB6D 5CDA21FC 617CF0D3 872D6F5E 7C2DD2D3 B73C04B1 2E783AF2
           6CC3A0C5

           CD5286DD 2A2EE884 37546EC4 7DAF7835 CD895897 D2018ECA F01D72F0
           E01E9521

           D4860053 D39092ED DAF86262 5F6DA22D A626C072 46F4A3F0 AFEA13A4
           B0A5018C
```

```
          D2AD47C1 FE16E514 8B571A8A 9E26909B 40ADC8BD 25F0E412 C36ED0B3
          31E05A3C

          D273E6C4 FEC08241 42E469D2 164547A9 61CEBE01 3D4E71CE 33CFE030
          BD32B135

r       = 2BA4341C 69516927 42D941F0 15750916 B7843EBE ECC2985F D00F2F4F
```

F.3.1.4 Verification process

```
t''     = 834763DE CD777788 F3112E05 30A21688 38D774D0 16D277F1 D6707D38
          57E3B652

          0F8C1BDE C2C29760 B4CE4566 D7D1A537 9AB5FF16 1643CB15 43DEC151
          D16AC5D1

          1516349B 5877F8AB 3AB768C7 62E82DFC 1E4FD282 B1735A9D 5958CA56
          A75EFEE0

          2FC0B8BA 2ED1B6D5 E5257021 C6AF6EAD 3AD57A4D 0209992C 7BB1F8E9
          35488563

          EC26FB95 94391FCE 715E9227 AAE8999C 2F6746E8 98722229 424ED0CF
          F10ECFA0

          EB42935B 580662CD FE4CC762 9CEE161B 0B018350 0629AF16 0D483659
          14460E62

          F9CA2238 B9D74EB0 E5172699 E50FF917 D69BE42F 121167D0 D79B4B75
          63CB39B4

          C5D3F35D DD1FE475 3A08F620 A7F37601 2589E335 2FD1FB63 0A600677
          20715311

c''     = 3251BBB8 815EBBD3 3A63DABE 3549A407 BB4E7CBA 95A4A6E3 DF9E2BAB
```

valid

F.3.2 Elliptic curve based domain parameters

F.3.2.1 Generation of domain parameters

```
hash function: SHA-256
curve name: secp256r1
curve p = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF
          FFFFFFFF
curve a = FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF
          FFFFFFFC
curve b = 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E
          27D2604B
curve q = FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2
          FC632551
g1      = (x = 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945
          D898C296,
             y = 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068
             37BF51F5)
g2        =  (x = 89002D14 40075619 83579EFA E861BEF3 37E3D8E5 C25A1958 3B63F332
             8BDA1B35,
             y = 44008920 053A6CFB BD54DAAF 26DF953D 243BD68A 197CF2AE D32F2BC2
             E48DBC7F)
```

F.3.2.2 Generation of signature key and verification key

```
x   = 6E3ECE49 1114DA5D D4C4CEA4 F76C36F5 BCE21695 32DC7E1B E2A9EEA8
      351093CC

y1  = (x = B41200EF B9D09585 5E62BFAB 9B7226DB C783778D C55B0C94 68C1B6A4
      B9F15A87,

      y = 34C4556C 5A994FD5 478E2E4B F40E0AE8 98B6BAE6 F8399699 36EE1920
      2749A8EC)

y2  = (x = A4EEA644 88140A72 BDF619DF A9D4E507 3A70173B D985EC02 3023B99D
      2C5B2DEB,

      y = 516F6A59 68602FD1 41EE116B 16CBFB18 B07557CA 6A3015EB 7950E3D3
      D2E8E9F9)
```

F.3.2.3 Blind signature process with partial disclosure

```
m     = 4d6573736167652065e6f742073686f776e20746f207369676e6572

info  = 5075626c6963207061727420 6f66206d657373616765

ω     = 869E0FE8 62276349 766DBFAE 1B87D96C F8F18C68 ADADA56E D4DA66B7
        599DA060

gM    = (x = 6873ACE7 4985B7B5 0D2B30BA ED4C4F39 0A931AB8 4559F7A3
        055DE584
        BAEA34F5,

        y = BB125309 1D9879D4 1865E77B EDEA0424 07C06736 DF2F8807 D2238AB2
        813CC430)

t′    = (x = 35A198D6 93F92801 0E08BD41 A072EE94 39013906 1E91B7A3
        8BF3BE86
        BADE2CFC,

        y = C863E557 A83BBC09 E136ABF1 CAB3EF20 EE76E709 20D652DB 2D356A0D
        7EB0F8E2)

λ     = 1C0B5642 3D96EDC1 99FDF6C4 D05D4626 44E486C6 966580D3 8722CCF5
        25106F28

μ     = EA84A814 9F813739 0C7A3D8C 09AC03F8 429FEFA5 C7868544 639A15CC
        AAF79FE9

gM    = (x = 6873ACE7 4985B7B5 0D2B30BA ED4C4F39 0A931AB8 4559F7A3
        055DE584
        BAEA34F5,

        y = BB125309 1D9879D4 1865E77B EDEA0424 07C06736 DF2F8807 D2238AB2
        813CC430)
```

```
yM    = (x = CBD355A8 FC5D1476 AACD7928 403FBEB9 4CE54256 56849271
        19CE2594 E8D138EA,

        y = 5EAE273B B96EFEFF 16867D37 527ED3F8 EBBC3666 264DE5FA CB95D63C
        2C1EF8CA)

tM    = (x = 08DDC4FD D5057144 CDCB4CE3 8CEB8541 52F52642 2605DDA3
        032286C6
        AB89FB18,

        y = 6FEC886A 8BEF3294 B468A5B5 3488E3F2 62354B4D 0668CD4A A494DCBE
        E6B36C66)

c     = 1969E191 94527901 D5DEF6AB 75B4A852 491D77E7 B3182FFE E1A394D1
        95C1487D

c’    = 2EE5397B F4D141C9 C964B91F 6C08A459 C36482EF 92A9493F 71C349C7
        E72CCDE5

r’    = 89BC02E9 44CB85FE 5AD3A1ED 57AFF040 746B0AB3 5CD50C66 9FB4513E
        72811449

t’    = (x = 35A198D6 93F92801 0E08BD41 A072EE94 39013906 1E91B7A3
        8BF3BE86
        BADE2CFC,

        y = C863E557 A83BBC09 E136ABF1 CAB3EF20 EE76E709 20D652DB 2D356A0D
        7EB0F8E2)

r     = A5C7592B 826273BF F4D198B2 280D3666 B94F9179 F33A8D3A 26D71E33
        97918371
```

F.3.2.4 Verification process

```
t’’  = (x = 08DDC4FD D5057144 CDCB4CE3 8CEB8541 52F52642 2605DDA3 032286C6
       AB89FB18,

       y = 6FEC886A 8BEF3294 B468A5B5 3488E3F2 62354B4D 0668CD4A A494DCBE
       E6B36C66)

c’’  = 1969E191 94527901 D5DEF6AB 75B4A852 491D77E7 B3182FFE E1A394D1
       95C1487D

signature valid
```

F.4 Mechanism 4

F.4.1 Finite field based domain parameters

F.4.1 contains numerical examples using the finite field domain parameters defined in Reference [24], section 2.1, identified by OID “1.3.6.1.4.1.311.75.1.1.1”, using SHA-256 as the hash-function *H*.

All values are in hexadecimal.

Values from 8.2.2

```
y0  = 2adb5089a512e7b9f329da09b28b505f85761a7c3f399ad257bb1b9250c53add
```

g_0 =

```
7d012e6556c9635e58899a4918636bf4d4984ef2c45e544b29930dd2240ab92793d-
2b5f905895be5a696848245b66afd38bfaa7f026b6bf32020bb6be8d4010eee70cb-
f52942c5c425c0dc4994f661d21a036385b74c1c5ffff2417bd33eb051c18d3de594d-
0b064086c7573a68dc91b7052faa972e0e3c7554eb39bceb642aa00a01331fd39af-
8c76a45aeaf24fe7ebd1710923a83060caf9f82636df0cea9de4e73c86190a92f2020f-
2540c2adfcb447b6b36666e6753db5e9ac844a088643217bdd5e6bc602f0f46ae-
2f72e1ced7fd60156c3403d686b592119e241415310c9a616ab3f4432efa18f6efcfd-
7123cfcb98959000bd611ae61f7b4f1de972d2
```

All values are in hexadecimal.

Values from 8.2.3

```
x1   = 3e4668267d6a6fe778ec3a189b384b44d029f3edc3532d618b88a729adaea673

x2   = af93c647ca51d4c950a616f6aa4cca9c3995589b0710783c3e3a513caf244772

x3   = 58f98bdb5985d501eac1de1057505c3782948c1b5949261d67cdeddf1bf49a5c

x4   = 1

x5   = 499602d2

xt   = 347e50f40edac4a1867f9d50827188324498d407a32945545bf9ef217eb23937

PI   = 50726f7665722069 6e666f726d6174696f6e206669656c642076616c7565
```

γ =

```
67106c2e235c854e033693c6f3c64736f2bab35b5a0a3c936f2bd77a1f7bb80f9e-
6b98274428a73378bfc6cab2a6c4c00842448c1053c8a27b198929e3f96b5d14ddee25b8c-
3c27f3519e0126a7439d4fcf1d5da0ed8f79f11cc8d7ebd709f265935845cf4169e5d-
cae9f6025f80ac15e196e9200525e29a2419539877ceeb4a4ecbcb93669ff37ca68bd9f-
082ca582ddc20b4f5b3a20144f9a20dc8e0ca77d118b3c74d014f44329f643e8396616d-
4e55479b471381ef67025d6701348f0aa6e61740659522f92aef68c1ce2a505a61cffde-
2578fee82066f52bb766403ca1b0b8632b9236a87ed26fbdc8148e1ca9c9f0102df0ed-
7610fc0f72d89587b9967
```

σz =

```
45d58f0e94743caa7a8580373761faf1dcf7e6784b4752c59ea2dc594dfe6601f-
597d5ea946f1d5023f0a51496555ec469993e699c8d92887f72869afa39d9638003fe-
f08648805ab1be60d2044de4287b7e26304f819e74e4b7c48131e488c36a7947f42e-
de8c58ea76d3b6219c8632ae765ec127fe16b3ec00f15ea6edc6c0c5a0d34f-
248cbd96d2c299bd4adfe65e026f2e209909502296b4efc8ae8a7a8088a01d-
4fb5ddd0fcf9636ed23b9c3029dc82e37011dd20c35fe235955c5cad1122061f-
92d5965e862120a3939aaa21db6250d818fde750782bf03af910364bd3c01321cc460b-
d88db307651ea5410eba24e9579489909a4d85c464b633150bba
```

```
w    = 240b649e253beb1fa0cb10324392ed90b516b437137749f1417e31eaf655692e
```

```
σa =

b412e85c2e4ffdcab10dafa22afa520c8fc2f3bbe3963ee839d1dfecb7621
548282d76ee11e49e8f85a967db12c5ee07f076a17a01634e926935b197e-
fe652ff9a0bd4903fad58f46f1f99fcc6c70c52604d341d4da182ee-
113199498a4908347065b21710b37c55c8361b8360791c876aec7d09f99f-
017ce4e367032cbb8efc777a89afac06efc4ce4546e150436487595f19d-
992fd5ee643a3a0583c36c7daf72714e349fdb2ad4ef077ef7e0b53821cc228bf-
d68a7da819fefeb0331381b2882114f9923d186bd0c6d4c655f439a07cc66c3c574b-
d2ee620a54a83e2ab99a9f0b2e1181d9c341451df7e977bf70bad01ca5b588f0f24e5ece-
f3afa8cefef3

σb =

9a10fec13c4b0795064b14d05d71e90dcd25cf72698ddf891e06896a6d-
2352d10d105d88ec13a6e4c99e1274c3997dbec80c8a9e3cd78a6388fe9b06de-
6aec7c7924460e30b9c47274602a789c8f2f4bf16a5159b4e639c3a1731d0e8b-
b88a7a50ec4cf2aa19e92269794256764b8cc51ec9d6e4a0e120ba4695f7a6630b-
c978045b996fd5f005e6d47eb7ce64146c18fcaa214ab5a5fe082842c8f0a-
0c60a478ac9b506195394a2261b379383be5c469f5a809867b61a7a10b637995dbbb-
b1fffbfcb6505f91b39b2de086fb0a1fd15b0be1485be281a38d6995d73c2366f3ab3-
e595fc6d66a81e465701cb6984a813718f3251d3f4ac199d1e9f2a5a8e3d20

α    = 270695120920c1308634c405c6310004303429095a98fa25b879d9c21b48ffc5

β1   = 62e24346ac466a03dc6aac189bd5c9d9ef6c1b6c792e1cec3338e5b355924089

β2   = 999bfb205b69f6f1c124539f37b832a0b2c046d305300acbc5d908f73fe84c4

h =

391f6372a5495a30f0980fcfb2eabbdb4a28ac9ef6eb6fabe2c559f9f-
1085cb4e97ebc24041d5c94327e888690d1e69bf1c4178c34dc0827755d2118da00a8f-
c3ee32d40b4f83af94ab9f781988506 0d29082e569850ab875ceaad8a6e469db-
75de05c1735d8431eab74f840749c1e62b7dc43a0fc1019951d5e79a43265d-
4ce8a508537d9ba807544e7d4e77c039821fdbae5fcc19071a8d2040104a5d01e-
0a1e2ade7e140430752ee9d39b12c7e5098f337743694120714a2ee682f95f-
742bec93f6129e2bb418a18f92a8fbcc900340185937d73d953c2e08511812c985e139-
ce82c43877f87e9fe96199a7e1e59348b39901536c060f269b800488c7aebd

α-1  = 4c2d9a2304ce5625b463ec3e5d038960f41bc765c9d2f987a94d5946e61bb21f

σ’z =

3ffd9fcc477673030edc05e77f97e46fb871c9e16e9f9d66f4b99e3c4a45f63beb73e9c8b-
bc44f1a05915758f057c3b13a9abd445a1b00c207709a091ab0ee00a7bafda057c4a962b-
f5fd7cd4dd6382ab2e345ebb87d6a8b8eff3d40e23935bd0c6199767977e40ba9f15c98
3a073ecec85fe37523b569710425963acbd6afaa057d62a0ccf4073750c80ba7ff4b1b-
99d5eb78f6d4ca835767f48b48b4660a5b34d91e9fbfa01f9bde5497badb688d4dddf-
cfb913841e39d8713b40798a5ee6711cd70b9492e3e09422fb6236dc19c9769b7e-
be9adc54123fbe920e28c9e3c69a008615b72a568c4fc15036f9b60a6f2fb35299eea-
349ba5aaae601a1460bc16
```

```
σ’a =

1ca98b663fb119f75694f4974e302183251431164b6d9631ec45ea0726709f-
38bc5e0e78d8d4fc99127a251ca2a7994280b824a37b4351baaca135b13ee4ad-
f09eeafd920be00300a36430f0842a934c8d73050821a440c84b3a41c3d95a952f
4e3304d4557fcf6ef1b7c45b843f07e648ec53bbe3c5fb7ea84c6df01c7674a00d-
b8710a4197137ab955e2fb89947cdf0f141fa969f0093c74208d0a1b60d-
d390739ee82ea9d3862cba5bc9d3dafedc3b46a277488c8320d5b4ca79b2e5be48d6dd70d
2af6dbc107e9c3c7f881a816fc6b0881b2134f39cf74ac06ba76d6cbbcc32c7ac7e7fa6d-
1d942202d71cda087a8c61c1b677246a5139fdafd1996f523b

σ’b =

c3ebe8ba7de9882982b59d43a2632c7564e37d2d9153df33cd1259fe6c7c21fbbd-
626bc7ef2dc7f490ef9153e3209bb417f6d3792d104935c5ad6ee901e926341975l1
3cbd670312ddc97b2c04a6f883878ccba63c39ba173b884e3647248a3154cd88a38b-
c7226349bb4fba25f2be591b4bd312959bc04fe22a8960ea9e1e0d8ab6c69455b544cfb-
71b58c9655f3e3fece016f1fcb4a64a73d19479eb0eea1db01ab3797695b2b61abc6a7ce-
66b9f4cea0809dad6841ab0226640736ec9096ae71fb69cbcf0876a6365dc6793086eb-
291da1e2026b7758905a79504580736c2b7e7a79dc7fef77530f00f3acaa3c0b-
170c7e077f13929b1bb1a57494aae7a3d

σ’c  = 2e96fc424078be020fd5943181381818bfc6704e34a2dcfd4908e85f0911bb20

σc   = 91793f88ecbf2805ec40404a1d0de1f2af328bbaadd0f9e97c41ce125ea3fba9

σr   = c5bcfd82593e59f1a1af91c044e80089a05d80870251797b8bd19137d8247dc8

σ’r  = 65f6ca041638047b8fabe8cd62cf4f20d32d1c37c3aa9a13f369ee25e6ada07
```

Value from 8.2.4

```
D     = {2,5}

U     = {1,3,4}

m     = 5665726966696572554944 2b72616e646f6d2064617461

md    = 446972656374206d657373616765

w0    = 2923450ac482b6a654924ba0d175978026f909d2dba3a316bfdc81cd35a3276a

w1    = 2c137d326d072627cc10485370316250bf924d67acda6e86802e4ae701e871b0

w3    = 6be173bdf765131fd2e206ca5a4fa58d7a300ea256bfbd51a5f6b23ebc3e26e3

w4    = 9f98e88fbda458597e2d95c9d4a32c3088bc6dd0dace59ab23221801b3d46293

a     = 40a9dac900d9a5c95ca6e0ee72af98b10513ab70e5fd7c4d5b8a965e22cfbdbe

UIDt  = 1916b2f56e8aa62ddadfdcfa7c87f2bd0880152ab4037f34a1b3758b328438b9

cp    = 25ff6f3ab2dbf1876522c0dc07c9a58bf10284b15d155fa5a9d4b949bd561bbc

c     = c716e004f05d96b762da801bd401271cd50ac6b9b7254c9d9cae8dc0e1765d6f

r0    = 488391a102c118454ed530a5fc8de64671dd8dbc9370cbbeb760357d020c8c28

r1    = 50beabfe34e519e972eea2dfc8bdef53b5f00c2a8c509bdb394cc6dbd3372518

r3    = 6fe395e7580b404e9a9c2a3029b00bc91073b8273b675ad55410e52efff030cd

r4    = a179591eead83abb201a2e1b62d893d552085a47da12dd948f6c0d25c0162da9
```

F.4.2 Elliptic curve based domain parameters

F.4.2 contains numerical examples using the elliptic curve domain parameters defined in Reference [24], section 2.2, identified by OID “1.3.6.1.4.1.311.75.1.2.1”, using SHA-256 as the hash-function *H*.

All values are in hexadecimal.

Value from 8.2.2

```
y0   = a6aba74b82f70f5fbc6366442fa8fa8dba7af900841fa4d3030cbba57526f3e
g0.x = 29fb21eec2ca3b81e5e8261debe078afc6b8ceb0e55d3a6a5fb463e9ca9bf9c2
g0.y = 6d3963868d3b7f0555e6fd8789c1e332cd2820e22934e7b5312cba80a074ff4e
```

Value from 8.2.3

```
x1     = 3e4668267d6a6fe778ec3a189b384b44d029f3edc3532d618b88a729adaea673
x2     = af93c647ca51d4c950a616f6aa4cca9c3995589b0710783c3e3a513caf244772
x3     = 58f98bdb5985d501eac1de1057505c3782948c1b5949261d67cdeddf1bf49a5c
x4     = 1
x5     = 499602d2
xt     = 737e093c37e7ce3da686d4ef42f7663da6f16e49eb718c29b1736f8e8ed12c7b
PI     = 50726f7665722069 6e666f726d6174696f6e206669656c642076616c7565
γ.x    = 9f7d798e68b8f58dc84b0ccbfd07c088f8d0fd68ba61a28bd9924ab9d5e53b89
γ.y    = ab3fd9346277deb4fdfbd4cf40cbb37f3f90b6960d419508fb1249e2c89bcfc1
σz.x   = b661e7e747d912e456e1b6536e682e4b57bb31906f6de0d06a6ce1809720963c
σz.y   = be542941febcb7957a169a4bea41cb221d2a44a2c1b003e80788781c4bb276db
w      = 3a938308c8b73a93883df4b440fe9d692b084b0d2b8eb1c8706c438763b69da8
σa.x   = 58b27f3183e89943d898e8e273b7e464d7d03c88d8f8a58e2b2708cacdbbc5f6
σa.y   = a6bdd5b8caa59a39b052db325c69740256184b0525fc058f238e4dec74dc45fb
σb.x   = 8b8ddb541070bb4f5805e33b0464963e864edaeeb7ca350e7bbb4e97a302c5c4
σb.y   = fe30399f487cba7c191d3a7d08507912173e74c45b39f5e9657b486403d747cd
α      = 56f729ae7786df236c1c08cb4d450d3293618e4f066112ace2ba975c73b22fd1
β1     = 9f4b5d48d4eef2a42928a00f85e67a2a5f11f401274ea1f4e47cccbcef83afba
β2     = ec362b01e8c45da46fea26dec10326fc406dfc62bd2eaa51aa686357223 6b5a6
h.x    = bab28428a4fcdac09f489b8a60ac464acbc658bc9bb3d9b76ceebbb9aaca6c0c
h.y    = 64cb93c0c508dc8bc5a84d47ee52afade1f57f4047000f9bfc0262b26da064f
α−1    = 74cff87d69124a6b0f9b7a754cb199054841cf156edafebb8a79624f0aeee1d1
σ’z.x  = b7307306b0710e153c0040239b03e3ac72ee0b4c09fe7431bf230d841aa7ac36
σ’z.y  = 5fc3cf6eaa31dae0b8eee9a4984c84fd2d7248f5b54b62b3fd089adea547f008
σ’a.x  = 6fe4049ec212765b219d7925e9fba1b8769641e5a2d8cc7d3afaad7061bac830
```

```
σ′a.y = efe335d7759ba9a2e0fa11949e1f5565ddca6d4e09496cc6987f143a1faac91b
σ′b.x = 82ffd18b249e58b677bc1076d90c5bec5bc6524f60ae6407cb6885b871f7aa89
σ′b.y = 90073801c10de596b2b9e1064a2185432fad755552e8d2e460c03fe01cd030a3
σ′c   = 6391255cd7aafe8f11866f4eb81326cefa0350b1f06c028a0209ac16a2a9eba2
σc    = 2dc82a6ac99f1323aaf0f5e3df9a0f99c2e4a0570a305f9f2ccae1095ca760b
σr    = e78e209c2c59dd3b9ffb176bb7809ac440dd0bf015a14ca0fe0f657681fe1a21
σ′r   = d3c44b9f151e3adf0fe53e4a7883c1c0c4640da52bb8586db4bdfe0aa7d1aa76
```

Value from 8.2.4

```
D    = {2,5}
U    = {1,3,4}
m    = 566572696669657255494 42b72616e646f6d2064617461
md   = 446972656374206d657373616765
w0   = 78e6234fba78429bb450923d27c233e156d07b81864dfcbe8cd9577f60058138
w1   = 348066dadfd741c72b61ad6d9b6c29e734810151ba331f2aea65c3e021c23aae
w3   = ce5a08a75b59027f8fb456259f8e221fb06f4adf042f7d01613cef7a1460a568
w4   = 27ff9e4164818cdb7f82c205dcf98a5b42a330b2775aa99edc07461f69876b2e
a    = cc7e6606fc61063b92e8d0eaa7dbb0942f99ad02af355df01ba9d56b1fd58333
UIDt = c9a4c12c656ab5fb3134d14d48d1020354c5f17d2258fdc4c65e57673ecc24dc
cp   = 0ee624e85271137640fa27fc1039c0326f7943ae0f963e88d6b3d4da8ced7d49
c    = da609b238aed949ba91ef469dadd20602f1f8bdafdbc52824caaf8eb920e851f
r0   = a9297d8e3eb3e788c83283de11544546c92c04d54b09b056f6545e5d7274e866
r1   = b649f1ed298fac8040d9d10972c9d6f90309227678dcf9c1c9ccd9d7e6e15fe8
r3   = 86c33e1156b947789e23a969017f3680f2b53d9f60afee5296f5d3cdb1e2fb95
r4   = 4d9f031cd993f840d663cd9c021c69fad06a9f8520b5f5a1831617f6d3dc0b60
```

F.5 Mechanism 5

F.5.1 Finite field based domain parameters

F.5.1.1 Domain parameters:

```
hash function: SHA-256
p   = EA8E526D A73D95E6 B7BFF5CE 62DADB0E 77EF2011 18E05B7B 804B3D90 9D41FA39
      D5DF3CE7 36B2AEA5 8480543C 6D187193 426D4996 15191D1A E3CE1949 0299F473
      3C4B93DB 85E91C61 353E4EB3 A483DADB 28268856 70A175C5 3A792E6E FB351A80
      11F08FF4 7754E153 37EB8B3F 928D5E89 0CE25758 1E26C963 A8CE38A8 91BBA962
      BB1F80C4 37C212C2 ADD6E1F3 294EE66D E35188B3 7EF7C183 0EB9CDFE 470A6094
      9D1191F4 04D6E14C B11B8DB9 5D43F7F3 354FE7CA FBBFE6E5 DF648FEB 711CB761
```

```
         09FDE8DC 17CA229C F2F318AC 2A2E318F 9A9FA539 08DCFAC3 D6990A1C 8E5A91D7
         BAFCCD3A 94492E0B DE9E7FCB F9BC9EA6 EFBEF0B5 6D8BFE7D 2F1B447E 07675CF9
q    =   D3FD9625 302930D4 93F92545 BD863E5C A5AB44E1 252F231A 7AF22059
b    =   3582862A 6057A0B0 1E82BB10 A1BCCD2B C12B5EF6 249726AE 1C43789A AB9F1BF7
         C6177F3D 0F43BDF5 522E5C08 D216B9DE 1416F457 7C99D19A 59DEC17D 3EE28061
         4E05A925 3274F05B 4FAD80CE 203D036B 519EBAAB 0B24AEDF 233D71C5 8F0DA63E
         A989A8E7 115562D2 FCC4F474 8AC2CC31 A752F483 23F5F93C A5E14C4C 606CDC6F
         9AF0151F 0D2A8E79 A7B9B7EC 5F43297A 64FF757E C180ACE7 DF4404F9 581D5812
         568F1C68 65E3360F 2EBDF001 48E05AC4 C2D3445B 0C3A3493 2FE59394 380BCD84
         E4A0551B 90544F0B 74B4829A 53388DF3 37AD87E3 12D27B5E B1AADF55 B277D7DF
         A27A74B7 843A7321 2BDC03AD 493E7C46 A7116F7D E076512A 0BE6FC8C B30AC005
g    =   538E954D 07553979 25438400 34D940BC 4561D32D 420495B5 1D3DAAF7 F1A9EBB6
         4F5EC7DB 4951B73B 005574AD D043CB57 B10FC2AF 66B84F79 DB1A6932 B273CB83
         67CAD584 9F812012 83C5AF83 5CDEC54A 348FFA8D 62B7BCBD 46649622 235A591E
         DD62C320 56C02291 3C6FEE11 6EAFBB0C 565F7F9F 2C283B38 B479F45F BDD0050B
         AB41EF10 BEAA2D8A FC06D542 5AF7ABF3 5F45A5B6 2AFC7AAC 67C1BCE0 E23EA069
         3D9C427D FC3D6D7E D569C6AA AA1E6F4B 888A3050 B08E6BB8 78AA6A8A 3C878593
         A6848B30 26DD3699 94E59503 6E1DFC11 BED8E8D8 A60C34CE 060689C9 8DFF0179
         6A7D1739 6F3C5BD5 EA570AC2 0D251717 C0726501 C1F12B10 0092F6C1 195C41E8
g1   =   A01A50E5 2721C0EB DF25DA11 96D2F53B 199B472C A455B931 C8636C6C 82808F41
         E21337F5 9AAB3F42 A53953E7 DF8372E0 8B805654 2A057F41 532A3730 B7AFF7D2
         CACD96DD 4C39E191 CDFB39AF 11BD99E0 75093F15 B86F07D2 508BFB3E 7FDDED96
         24A0899B 702C1CE1 F97D7471 6E976383 4061437F EF83BFA6 F9166DBD 0AD227BA
         F2604E6E 7840060E 49C06676 947CF524 909DB435 DA7F72C3 94381E9E F4C48384
         8C6CF85C 419C23D8 7DC071A5 7E0798B3 21F61ED8 1BAD405C B48AA55D 56121334
         3750368A 9014D9E3 E92C079F F36E657B 31F8493E 34598F85 F71E01C7 44567AAF
         280185E4 5D73B984 689DC02F D7F420A7 97688B4D D3969CF6 D3D5A79C B668DAD6
g2   =   C95622B2 DDB19CDD C69F7457 49B14DF4 2289E62D 98259E5B AA471D15 9C493B8E
         70A01BEB 303131BA 98FF6B18 DA1A1916 BEA2C61D E4EEDE24 DE83B9D5 E5F00518
         83FC6A3C CC2CA57F 4D4CD7BE 8CDC0F5B 2AE38FFF 86EC85EB B8AE1A6C EA9019EF
         A740E426 A3AD0706 55CE8943 10B1BB99 FC5E940F 80212FA9 7825CAE7 EB0EE842
         7AA24388 F4B55CEE 4FB7A23E 7A887757 8F18CBB4 5E4B9EA3 96691E60 6C57BEBE
         B6ED1511 6E54193C D639FAC7 EABE6C7B 22C7BE99 FA9F3192 3B64DF19 45030134
         599E4AF5 F0117A28 2A04CA61 B510ECA9 3E8CB4BF DA6B361E DEC5EC18 2423F0B9
         5DE5B509 9727E0C8 16634DB1 2F920DDA EB9CEDB1 A557B77B 13752FA6 10892046
```

F.5.1.1.1 Generation of the requestor tracing key and the public requestor tracing key

x_{RT} = A0B489A0 4C1D5B00 996FEAD2 5DAE4590 6391C029 A105159E 22B0D86D

y_{RT} = 70C39235 4931918F A306BFF9 F0DC9DE1 54CE6EDB 6BB1ECE2 E795407C 94B5C4F5

FE1B5831 CD425CC3 4200E270 DCA609DB F13C0DE5 6135E94F 0D093E4E 98B07861

F7A2ABDE 280E8A62 75E52C6F 5D591FB3 1059C1DF 1504D2FF D51A06A9 B2EDFB2E

7518C041 7D8FEA80 525803F4 B1499558 50633771 13AA08F2 893BA451
3A359F1E

05D7BFCF 887F982B 6482540F A1B68553 F4A1B94C CF8FA381 08ACCE14
0C605C75

4A33636F 234F65B7 B7704BF8 34104E05 EBA38377 3B6C6200 76D3D4C4
934B90A4

607F4E44 88F805A4 1DBB9F60 A87C5A4F 03205E5A B2437098 CEA02CDB
B3CB2261

459C9006 5AF125F9 1AD4F20B 86E1386C 7E3B9A04 16E0F311 A4F12916
1976CFD1

F.5.1.1.2 Generation of the signature tracing key and the public signature tracing key

x_{ST} = A57B8EC6 AB7FA22A D472D635 8F89DE36 55440AA5 0E5D2061 2BF068A2

y_{ST} = 46F1A0F7 8EB4AC9B E6B466E4 91ACEFEF 2A3AEBDC FEA102A4 D0677F26
590BAF0D

E56DB6C5 EE6B0FB8 65E9CF8F E3B76355 D4BF9DFF 483650DE 71D7695A
A6C34907

4B8B5FD2 C19D8062 734F593F 4671FF27 2505E08B DB091522 BEE1AB10
59AC4809

FDD259CB 9C82ECC5 0971B47F 32DD49D7 3CBE5E59 A767E289 5D987B24
AA7B3402

6AF2734E 95C21917 50B73E79 C764F3C5 5B5A4F0F 078479AB C998EFD9
A2F287B7

372524CB ED43B5EF D774070B C13E5C7F 77EBEA01 FBA37FD4 803AF6D7
2A791116

7296B643 1266E02D C2332338 A736FB34 5CF400AA F921DB8E 8E86E8D9
D6BD1BB3

E73693C2 8B05FE89 D0432D4B 9AFB2B81 4F2D3393 EE39EF7A B3B33E12
AE397567

x = D24A8258 6444B033 02E7A739 1989FAA4 B53F7D82 1FDAA85C 231E73C0

y = B5DEF074 81FCEC05 CBF75BEE F6B2F7DF 79EDFA71 E8EE6E32 08CC7023
FECED305

10C77E29 C4CC0D24 68236B3B 0011D88F 2675A8F3 71D25F47 CE2374CC
2FCD5D85

107B88C3 37076ADC 30D7F38A C62CFFC8 A676C82E 0EB34C2C 4C279A87
82B72B83

298BB221 FBA92615 BCE91A21 4F746B2A D4E169FB DA464E88 AB0A030E
C89EFF9F

7E5D1365 C1150817 5D2B13F6 55C21C0E 9AF51592 2DFEAAB1 2D96A625
F3A23BEA

```
        18A9AD84 BBEC3E3D 4C49C4F1 B7B2F914 D5EBF6FC A662A3E8 2EC8F7E8
        346C8BB2

        039B54E0 B9C7F552 7590560F E2F47CB6 C7C68731 BACEE113 F81E4B64
        E3C8EDB2

        E6D07C31 2004E178 A43BCEB8 6268636A 902FD69D 4EAEAF6A 05CD4E70
        E9C5DB42
```

F.5.1.1.3 Generation of the private and public requestor keys

```
x_R  = C014BF98 85FC412F 693F874B C0747371 1F936C9C 2F8162A3 F59DB680

P_R  = 79A28FF7 752BE047 5D20A9C4 F503C5B5 D2F59271 3473D858 ED4C9B99
       04490A1A

       FDDFFC5D 6D4BA63F 90810438 79988A00 130DAE2E 6D823131 99837957
       4DA6BA26

       9E963872 DEA8B147 F91C2653 4022A9C9 C39DB839 0A7BEC23 56114397
       124FB3CA

       386317BC 5AEBD2A8 87764C0E 87982133 E7D0AC97 00971248 3ECB9C47
       27BA2BE8

       34CFFF0E E2DA87F7 85BA1262 D6BC66BD 0B1E6AEE 1A96B4BE F6EC1CD4
       81883089

       7ACAD75A 7A4ED9AE 699FD86C E29852D3 3377E4E6 0569C59B 78A33DE8
       22ECCE8D

       8820F62F DF3045A7 6281D22A 6C5A3CA3 B5B946CF A8299CA9 3A4EF596
       D63B4301

       D9669E31 ABC0829B 9DED7950 A9026F85 32FB76D1 41884FDD 9AF7B191
       A59EC820
```

F.5.1.2 Traceable blind signature process

```
m     = 4D657373 61676520 746F2062 6520626C 696E646C 79207369 676E6564

s     = B568B8A3 CD05BE09 2B66119E 8E9F7C82 F2E19E97 00DF1EB5 ED8AA1A2

t     = 58CD7309 3260FACC EB647F4F 372B9CEF 0D046533 011589B4 D7C6EFB1

α     = 46C27258 7AAEEDC4 C9555306 E8EC87DE E235C58C 9F1DB54C E7E03328

β     = BAE2BEC6 E24749F9 F22DB4E3 15420D8A 3438E9DE 36705B9E 4B947027

γ     = A3992824 830A3611 2A19F9CA D8290449 D71F55A4 5D9CFA68 B397CF77

E'_1  = 6C2CEE53 3B177776 25DDA008 D24C943C 08187F2F 99284094 48A72249
        ADA169F0

        5A302A82 64F11E9F DC8418A5 B6D7A4FC 24BBD390 DC8B37A0 47118BA9
        40CC20A5

        7DA8B5E3 356C21F3 141824D1 EDCBAF77 8B55B632 67A3EE0F 1D8848F1
        623D89CB

        D2964EFA E0EBF966 EC087614 1CA291E0 A321FCBE 12D7D713 F4CF17E7
        DA971F24
```

```
          084721C7 DC62C98A C42735E4 92FDE764 741611BD 9929B0BE FF15EE33
          E45A1F10

          492EB3E3 BC2C0529 EE86CD85 42B7C8C3 0EB24791 C7FCF7CC 416489D5
          347C435C

          1CBDB941 31459A08 84CF1EB1 F8F1FCD8 7E745346 937C8596 BFC51B45
          E26F20EB

          47D22A27 B6C43D19 75774E68 23F898C4 281901F3 B4422427 ABC87D91
          92EA0BE8

E_2     = 1E8CE448 C7BA4E92 406B86AD 1F407600 26D529FE 8D888DD5 075087C0
          C610F48F

          DBC5B827 564D7E78 470340AE 0B0EDBA2 709FC565 FF3701E8 65F0F650
          389F03AE

          417FDBEA CC8ADD2A 1788D84E DFF3BB4B 668C733A 95606A77 C173ACA3
          09B0F290

          7DE9BE39 F03BEAF7 D7E37902 6457F230 0A51E466 FFACDFFC 8B648290
          0EFA056F

          F24793FE 3ECF9AAD 65C7CB5B C48F48C1 965DC106 1B362400 5118F0D7
          6BE11198

          F613092B A6F6EFFC A96A3805 9B5823FE B143BC59 E81BA586 84DD7ABC
          3100A8DB

          5C9577A9 3E1BD938 AE03AA28 CD107A63 C814307B 65707BF3 A57E43FA
          87A1134A

          1EE56639 532A97E8 27F1E650 890F9DE0 06765C6F D26E8C21 429D224D
          EFF1B1C4

E_1     = 0483547B 53647FD0 FE4FC369 0411FE12 AC900600 B323852C 9AF12C0C
          30D5AD00

          264C4566 4E56427B AE974471 5552D752 29A218D0 DF0F5254 960F15A8
          994198C9

          6866D039 47F9704D 9D532CF0 3366C2F5 6981FAEE A907D17A AEA15EFA
          1B3CC753

          37E867DB 8E29C02F 4C62ED75 C59711EB 9CC08C4D 5F6DC5E8 12F4BB2D
          8B0BADAE

          A9A8565F A30F9A41 53809CBD 784610B6 2C49CA5D 082C8E30 020AD996
          E0B81563

          25DE78BA F644E0BE CBDD8F0F 5F95A6FE DDD8FCA5 67F0938A CBCA426A
          908D86E7

          DD280CF2 BB7FC8F8 9521AB88 2E415652 83A2AE47 E868A07C 7E5C173D
          5AB0E40A

          8E06CF0E 76F818D8 F65FDA6B 1A7328C4 F90B1C4E D5E53F4E C4CAA1A0
          767EDB61

T_1     = 8715AB6C 07AB2E18 4630FECF 16EE5330 9CBA5DF9 72921570 6355B159
          107DEC74
```

```
           525DBE0E 3D7C2C1A C690EE44 8DC6CE0B A89089D0 505B0A9C 6D2DBA0C
           99B88386

           A1B8BDC0 08099065 3E041A36 9DD1A2F9 A90AF826 F62BF0B0 F70D016C
           85F54256

           A3696359 16E25D16 41547D20 3A1CE72A B2F8547C C3B3A5A2 A9C48700
           C42C994F

           A522234E 574B99D7 FDE478DB 755D682A 6FF55B74 B89FB0BA 30C5FE91
           474114BF

           4DEE1180 8F2AEA4F EAF5CEC7 A58CAC8E 41B4589B 17A55E00 910EB9E1
           B4E861DC

           6AC2C14F 03D48978 28845A64 40EB4D7B E2A48748 1CFEBCD7 245552C5
           A456D5A3

           D4AB483F 51A61057 1ECD1BE6 F5228CFD CE0F92D0 E51B7043 C13984FC
           ECC46276

T_2      = D258D7AE 60A810C6 F361897F B5959B46 56D6D3E0 8AA371B5 6DDFBA52
           2FF28D9D

           B9B04257 EE605959 12831A38 93E76BC4 89FFCA50 822F1539 C587AA42
           775DC617

           3C8339C7 064C4299 33C222E7 7E8AFD88 F581A55E 3F2EB763 1A886E80
           40A0E727

           25ED6375 B7AFE72F B492D183 F5F6C743 226CB612 D486E83F BE328978
           2427896A

           BE63F121 7EA4B679 88C35B50 2C0770FB 04BCD018 32BC5DB3 C6B1C798
           41B3874A

           8FAC2072 A7DB7681 00935A6A E0C83269 398589E4 6CF9E671 1DB924D8
           C0248507

           CC1A7288 FB139D62 92A132CB A51CA076 F7C6836E 8D1C32FE 90E09B92
           2CFED858

           1FBFDDF6 4DFBD54F CF1A3565 DA6BC129 F4FBC148 6A445C36 AA54D39C
           43FEDDB6

c1       = 4A403DDB 03BE4C4B A08056D6 27408390 8277F886 65AAFD5E C2E770CB

r1       = 1802C880 D0A8FCFF 4CC54D71 BD4706C0 9159687B 3A8E6F67 E25F726E

r2       = A79D1A86 C01A5931 BF695CA1 4D0DB15E FC7EADBC 550F7DB2 B5426FBD

r3       = 30A77859 AD9EE030 89D34D66 3BD2C2A3 0E21DF48 A2AE8FA7 8D1B74AA

u        = 301EC01E AFE7785A 407E5FAF EB904429 96875AC4 14F9017D 77F37972

v        = 64C6CDA7 B3190EEA DE5811FD AE3C45FF 9E3950CA DDE9C1F0 223A79A2

η        = 6A6BE642 6663E711 5D6E2088 DCEF7AC6 98B141C4 8D6696B9 CF7CDC43

δ        = 05405EA7 6DABAA60 6F164962 8F70A0E6 23DC1FCE 47E409FC 3B53407A

E1”      = 0483547B 53647FD0 FE4FC369 0411FE12 AC900600 B323852C 9AF12C0C
           30D5AD00
```

```
         264C4566 4E56427B AE974471 5552D752 29A218D0 DF0F5254 960F15A8
         994198C9

         6866D039 47F9704D 9D532CF0 3366C2F5 6981FAEE A907D17A AEA15EFA
         1B3CC753

         37E867DB 8E29C02F 4C62ED75 C59711EB 9CC08C4D 5F6DC5E8 12F4BB2D
         8B0BADAE

         A9A8565F A30F9A41 53809CBD 784610B6 2C49CA5D 082C8E30 020AD996
         E0B81563

         25DE78BA F644E0BE CBDD8F0F 5F95A6FE DDD8FCA5 67F0938A CBCA426A
         908D86E7

         DD280CF2 BB7FC8F8 9521AB88 2E415652 83A2AE47 E868A07C 7E5C173D
         5AB0E40A

         8E06CF0E 76F818D8 F65FDA6B 1A7328C4 F90B1C4E D5E53F4E C4CAA1A0
         767EDB61

T1'    = 8715AB6C 07AB2E18 4630FECF 16EE5330 9CBA5DF9 72921570 6355B159
         107DEC74

         525DBE0E 3D7C2C1A C690EE44 8DC6CE0B A89089D0 505B0A9C 6D2DBA0C
         99B88386

         A1B8BDC0 08099065 3E041A36 9DD1A2F9 A90AF826 F62BF0B0 F70D016C
         85F54256

         A3696359 16E25D16 41547D20 3A1CE72A B2F8547C C3B3A5A2 A9C48700
         C42C994F

         A522234E 574B99D7 FDE478DB 755D682A 6FF55B74 B89FB0BA 30C5FE91
         474114BF

         4DEE1180 8F2AEA4F EAF5CEC7 A58CAC8E 41B4589B 17A55E00 910EB9E1
         B4E861DC

         6AC2C14F 03D48978 28845A64 40EB4D7B E2A48748 1CFEBCD7 245552C5
         A456D5A3

         D4AB483F 51A61057 1ECD1BE6 F5228CFD CE0F92D0 E51B7043 C13984FC
         ECC46276

T2'    = D258D7AE 60A810C6 F361897F B5959B46 56D6D3E0 8AA371B5 6DDFBA52
         2FF28D9D

         B9B04257 EE605959 12831A38 93E76BC4 89FFCA50 822F1539 C587AA42
         775DC617

         3C8339C7 064C4299 33C222E7 7E8AFD88 F581A55E 3F2EB763 1A886E80
         40A0E727

         25ED6375 B7AFE72F B492D183 F5F6C743 226CB612 D486E83F BE328978
         2427896A

         BE63F121 7EA4B679 88C35B50 2C0770FB 04BCD018 32BC5DB3 C6B1C798
         41B3874A

         8FAC2072 A7DB7681 00935A6A E0C83269 398589E4 6CF9E671 1DB924D8
         C0248507
```

```
          CC1A7288 FB139D62 92A132CB A51CA076 F7C6836E 8D1C32FE 90E09B92
          2CFED858

          1FBFDDF6 4DFBD54F CF1A3565 DA6BC129 F4FBC148 6A445C36 AA54D39C
          43FEDDB6

c′      = 4A403DDB 03BE4C4B A08056D6 27408390 8277F886 65AAFD5E C2E770CB

ω       = A02A5465 436C9B0B 7850EEB0 C1B311D8 9B577343 CC851765 2DDE6F23

m0      = 0E9816F0 32B788FF 45D70ED0 B5D251BD FE86C3C8 1C55FEEE 76BD110A
          391D1831

          BDE1481D 560E380C D0CC0C61 35738DE4 B85D0E6E 097E2F61 08590701
          F675050A

          44B6D81A EBD510DA B6CA3FB7 B334B5EE 8B26C5A7 6D89AAAB 040359A3
          DAB142C6

          06BCE36D C0D88C37 15AA7AD9 2ADC8843 41BEABA3 56EBDB52 3E5892CF
          F2C84D94

          0217D18A A65BC40F 494EA319 B19881F5 3959A1B6 8FE742F9 73E4FF1A
          90A495B8

          776BCD73 6E6A8D38 594037A2 AD17C4E1 B34552F5 1CC75897 675A94BB
          5BEE0648

          F5852B74 2B373588 269B2AD7 C1D1D05F 16595BA7 8A20C981 13473841
          E4082C32

          4206FB2E 0107CCB0 656A2954 073814A0 06507A67 DC959641 7CFF0508
          FF5FD4A2

z0      = 0567C5C7 157F2857 EFAB48C5 C9422E90 0D92C623 550AE900 30695432
          9605C8CE

          4C3C64F3 F4CEF22C FDE144D0 55B37C24 C9A2A31D 6B95FE87 841AA731
          67B34483

          E3EB551E CF44562C EEF2B057 14AFBBE1 FDD73E18 64A76061 B211AB25
          C02866BD

          CF8C97DE 42FED2C3 CE2FA1EE CFF2E9AA DF3A09EC 9FEBBD91 4C0DAF4D
          2311F67E

          3A6C4760 765E81BE 8CF9C121 04F93BEA 9523F6F1 19003760 E45D91F7
          22723C98

          3FF94B16 5137D323 3B648B7A 93D3FB99 C96E01F5 6CF9969C 3D4F87CF
          98238CDA

          FBE56A1A 014EC8A8 A64CAEAF DEC01348 E4F1897B 3F2FA71F F4C176A3
          226801EB

          4704FC39 EBFF2F45 B0BDE00E 0845C342 568D4DA3 91E28FC2 82DCF6AB
          2EADF143

A0      = E97B93FA 40A1B792 57E64495 2DBCFD8D 475A8454 FAB93596 9A2FEA18
          1978AFC1

          AEAC4438 07C421CB 03C2ADF2 70BB33DC CACFD2C7 FD356C05 85177840
          DA8A8028
```

```
         89514BC2 401F1F03 328E5084 232C80FF BDF44775 4DBFE4E0 11301301
         09D7DBD3

         BD339804 BFA50907 E195E395 03A97F5F F40589E1 C46CB3A2 1BA46E65
         3F3AA3CC

         673B5D06 0FD82DF0 0BC73E28 09888938 0B29E394 A9BAACC1 FB31D1F9
         9EB9EB7D

         54BA25EE 617F7E55 1503FE53 4E2D98E1 F8D4A004 5DDEC6F8 E4137563
         37252A8B

         1E7F3C71 E987C2AE D5C09AFE 1D7CAF73 1B46B7E1 B3AD3B07 1C326CD6
         21F2BF0E

         56EA7C52 BF005409 C5524BE9 127A5B97 02C8A73D 82B40FFC 9950F6BD
         AE0A1E48

B0     = 6C9A4004 C0FA08D0 B91A2F5F 6C775270 9E34C8ED 7C0AE696 FDA13FE3
         72B2CD63

         9101FC13 F31E3988 83664E6F A7552A37 0C982E06 15B00250 09C20DFF
         70493995

         0914EA93 E54296CC B67E1693 3D92EA55 6A02017C EB473CE2 43A0D5EE
         9232F797

         1B347480 828D4A30 9ACE28DE 04CAA782 1F20F558 D9400A06 45470CAE
         F5CCB1D5

         35D03586 41DC4D92 A1F3A900 6B222D94 31454F99 A48FBB6C 999EE733
         80FF6C43

         B02FED28 8381091D A05DE954 C7824C57 C5A02C73 A54EE851 437734E3
         EC6A026D

         3F6E239A C5B23A8F 5C9B79E2 AD8DE656 0DEFAD0C 9CEA2804 7161CBBA
         A48F9314

         87D29978 A7A00E98 D6275113 78525935 A5760889 4A1CEE3B 5BC3DDBB
         C2EED797

m0     = 0E9816F0 32B788FF 45D70ED0 B5D251BD FE86C3C8 1C55FEEE 76BD110A
         391D1831

         BDE1481D 560E380C D0CC0C61 35738DE4 B85D0E6E 097E2F61 08590701
         F675050A

         44B6D81A EBD510DA B6CA3FB7 B334B5EE 8B26C5A7 6D89AAAB 040359A3
         DAB142C6

         06BCE36D C0D88C37 15AA7AD9 2ADC8843 41BEABA3 56EBDB52 3E5892CF
         F2C84D94

         0217D18A A65BC40F 494EA319 B19881F5 3959A1B6 8FE742F9 73E4FF1A
         90A495B8

         776BCD73 6E6A8D38 594037A2 AD17C4E1 B34552F5 1CC75897 675A94BB
         5BEE0648

         F5852B74 2B373588 269B2AD7 C1D1D05F 16595BA7 8A20C981 13473841
         E4082C32
```

```
          4206FB2E 0107CCB0 656A2954 073814A0 06507A67 DC959641 7CFF0508
          FF5FD4A2

S_I     = 39BED59F F096F2E0 38CB8E77 6A2DB821 E7D202C2 40E97815 E15ABADD
          FD0E4FD1

          F8BF1598 C184A86A 9DB47EA8 B99C3832 5F496044 BBB2CC91 627004CE
          E03CC45B

          71B3AE18 FB955E73 97C8A2BF 0F74E625 841CCCFB 502D91C1 85138AFD
          6BBD4B02

          ED674548 F4E2CDB0 F78E1AC8 4F3C1A4F D03088C1 128B73A0 0EF91DEE
          89CAE472

          6DF686B9 5018495F 7941E8E1 BDFA9685 A58A383B A969D09E 149CE6CC
          3EE00F31

          2B432061 1F35FD62 A64175FA 923143C0 B7B31A6A 724DE347 3F21A99A
          C269BDE0

          1A6DFC13 99E268B5 C3794EE9 36206B72 8AB44D88 0EED8775 3B1DCB46
          E2B5C888

          3129C3DA 176FBD31 65B24082 252A54D4 AA752BCC BBD7BE8D 3CB0E4D6
          B9E09CD8

A       = 1A6BF9AC A013BCA7 A2D1FBBD 8FE0E36F E84083E0 BAF6CC4F 72EEAAF8
          8C9D9F2E

          14EA8829 1EA00C75 464821BD E0C69B2B 70E83EEA 514C63A4 A8EC70E5
          E5E01A7F

          4BF5B00C 05EE1BAE F982E76B D52D2EDF 6A53B412 3FB9235B 8B521EA5
          5B3CA076

          D4064576 43AAA6A1 166976D8 32237AEF C020FC99 2A1C5691 FA93EDDE
          1EBEFAB9

          531C1BEE A361DDFF CC2C49A9 93E18D0C 6D228D4D 74D6B37B 7448AD3E
          2849041E

          D633BC93 DFD7B7E1 D4501F42 05C7A599 4B45DE79 697B02FA EA71C289
          AB2E50C7

          5501AEB3 5FB70E7C CD4C6D61 C73CEA18 EDC5FF37 76FBF323 6CE3F67B
          9FB3A878

          51D91AFA C20ABF68 1D3A3C4B F7FEB6D9 79C1F06E 40977C9F 4C699A2B
          14C76139

A2      = BAACB27D 1966F09D C686F5D6 FEA51BB3 5B7AECD5 AD711D0B E4260D3D
          BDE2001F

          C9A0A287 782E0947 780E4AF1 BBE227D3 93A2CEEE B4B0AFE1 0F95C580
          93EEC11D

          27345E36 634A4B3F DB3A56DC 705F539D C444B613 3D43AE46 8FB96A13
          3F4917F4

          6F9A75DB 093EC70E FE191683 D7DB3D73 ADD1C66C 3136AECF DD21E857
          EAD4CF7C
          0EC457A7 FCC78A96 0C140D9F A935E50F 677ACD67 95F8F3D7 76F4B570
          E0301AC4

          98675120 CCD3B898 8C1EF7D9 02D9C569 7AA67BBD 4696FF53 A7E7BB2E
          B253E151

          A8591C5B 69AE7F09 0502B0D3 0DDFD48C B1A89EAA 72BBDDEB 2AB013A2
          13D1AD19

          D0DDB00D 85BE1135 C3D361C2 0D49494B 207D2213 F1F1D8DC B92A3C72
          85330ACB

B       = A02C6EA5 B154F861 6C65A433 B4509600 AC308894 81398C5B 1E31BB2E
          BEDEB6AE

          4EA4FD3A 9CD27758 2ECDF599 75AE9F5A 0FD7AC55 2679682A 605D8F22
          14E0B716

          E27E1064 1F1886C7 45511A86 8BA063DA 6DA190E0 72D61C9A E6DE6125
          8DD2A0D1

          65709CEA B3D3984A 545E597E F1B4BFAE 28B0EE4A 16A9FC95 EAE8A3E1
          04101E83

          118D0351 5BBE2B6D C5B63AF8 A18C88B3 97B78F93 E881306B 21E7B0D7
          2F83857E

          46E245CA 72773BA8 BDEC3039 43A14B8D 1BA9CDF2 7BCDDEB4 AF90D591
          9324DCC8

          985AA9B3 949AB57F 60802F34 A098A281 86A3C95F E5908F91 6F9DD5BE
          E4BCD7EC

          91F78164 6DE16315 C168C78B 1B84752A C0FA74DC 0D0FF3AC 334048A6
          CD6F7CAA

D       = E35E5D76 45916CB4 23C6F1F5 002AC060 0B98D987 B389D713 BD6AD422
          F365A562

          BBE93EF1 B413B854 96FF34AF 66281D7F A6F31C47 DACDD118 157C4DDC
          F3ACADAE

          D8639812 39BE45C5 89D10191 017DF7F2 742F7D41 4D88D341 5C15D281
          E0554231

          70DB1F0B 628C1B62 505FB310 C172B735 AAFE10DF 8B4BA016 2BB63D09
          6EBD8E3D

          F717BEFF 19772740 5D542BBE 0C3EF4EC B7B4B058 B3E92DF6 9FE30392
          B4CDCD0D

          7A8B7DD9 1F608848 AF2E50B5 F9A2F98A AB50012E B4623524 C4FEDEB7
          30AA1828

          BBD155F3 CF2615ED 149922D2 41CEBF42 E788249D 2869154A 2684BF5C
          C928C9CB

          4555D8BB F4906E20 14201C2B 875C495C A2AEA36C 1BE901F4 B87A6B22
          303EBD3B

E       = 85087217 012FE7CA 86F898AF 22111E1A E4C96B68 EA4B62D7 595BD9F6
          F06E0BF2
```