INTERNATIONAL STANDARD

ISO/IEC 20008-2

First edition
2013-11-15

Information technology — Security techniques — Anonymous digital signatures —

Part 2: Mechanisms using a group public key

Technologies de l'information — Techniques de sécurité — Signatures numériques anonymes —

Partie 2: Mécanismes utilisant une clé publique de groupe

Reference number
ISO/IEC 20008-2:2013(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 20008-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 20008 consists of the following parts, under the general title *Information technology — Security techniques — Anonymous digital signatures*:

— *Part 1: General*

— *Part 2: Mechanisms using a group public key*

Further parts may follow.

Introduction

Anonymous digital signature mechanisms are a special type of digital signature mechanism in which, given a digital signature, an unauthorized entity cannot discover the signer's identifier yet can verify that a legitimate signer has generated a valid signature.

ISO/IEC 20008 specifies anonymous digital signature mechanisms. ISO/IEC 20008-1 specifies principles and requirements for two categories of anonymous digital signatures mechanisms: signature mechanisms using a group public key, and signature mechanisms using multiple public keys. This part of ISO/IEC 20008 specifies a number of anonymous signature mechanisms of the first category.

Anonymous signature mechanisms of the first category can have capabilities for providing more information about the signer. Some have a linking capability, where two signatures signed by the same signer are linkable. Some have an opening capability, where the signature can be opened by a special entity to reveal the identity of the signer. Some have both linking and opening capabilities.

For each mechanism, the processes of opening, linking, and/or revocation are specified.

The mechanisms specified in this part of ISO/IEC 20008 use a collision-resistant hash-function. A hash-function specified in ISO/IEC 10118 is to be used.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this document may involve the use of patents.

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

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

— Electronics and Telecommunications Research Institute (ETRI)

  161, Gajeong-dong, Yuseong-gu, Daejeon, Korea

— NEC Corporation

  7-1, Shiba 5-chome, Minato-Ku, Toyko 108-8001, Japan

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

ISO (www.iso.org/patents) and IEC (http://patents.iec.ch) maintain on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

Information technology — Security techniques — Anonymous digital signatures —

Part 2: Mechanisms using a group public key

1 Scope

This part of ISO/IEC 20008 specifies anonymous digital signature mechanisms in which a verifier makes use of a group public key to verify a digital signature.

It provides

— a general description of an anonymous digital signature mechanism using a group public key, and

— a variety of mechanisms that provide such anonymous digital signatures.

For each mechanism, this part of ISO/IEC 20008 specifies

— the process for generating group member signature keys and a group public key,

— the process for producing signatures,

— the process for verifying signatures,

— the process for opening signatures (if the mechanism supports opening),

— the process for linking signatures (if the mechanism supports linking), and

— the process for revoking group members.

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*

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

ISO/IEC 18031, *Information technology — Security techniques — Random bit generation*

ISO/IEC 18032, *Information technology — Security techniques — Prime number generation*

ISO/IEC 20008-1, *Information technology — Security techniques — Anonymous digital signatures — Part 1: General*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 20008-1 and the following apply.

3.1
assistant signer
entity that can help a principal signer to create anonymous signatures, but that cannot generate anonymous signatures unaided

3.2
member-list
list that includes the identities of group members together with their corresponding group membership credentials

3.3
principal signer
entity which is in possession of a group member signature key and which can create anonymous signatures using this key

3.4
secret seed value
secret data known to a group member and used for deriving group member private keys

3.5
security parameters
variables that determine the security strength of a mechanism

4 Symbols (and abbreviated terms)

For the purposes of this part of ISO/IEC 20008, the following symbols and abbreviations apply.

bsn	Linking base, either a special symbol $\perp$ or an arbitrary string.
e	A bilinear map function $e: G_1 \times G_2 \rightarrow G_T$ such that for all $P \in G_1$, $Q \in G_2$, and all positive integers a, b, the equation $e([a]P, [b]Q) = e(P, Q)^{ab}$ holds. This function is also called a pairing function.
gcd(a, b)	The greatest common divisor of the integers a and b.
G_1	An additive cyclic group of order p over an elliptic curve.
G_2	An additive cyclic group of order p over an elliptic curve.
G_T	A multiplicative cyclic group of order p.
H	A cryptographic hash-function.
m	Message to be signed.
n	An RSA modulus where $n = pq$.
O_E	The elliptic curve point at infinity.
p	A prime number.
P_1	Generator of G_1.
P_2	Generator of G_2.
q	A prime number.
Q_1+Q_2	The elliptic curve sum of points Q_1 and Q_2.
QR(n)	The group of quadratic residues modulo n.

Z_n^*	The multiplicative group of invertible elements in Z_n.
Z_p	The set of integers in [0, p-1].
Z_p^*	The set of integers in [1, p-1].
$(a\|p)$	The Legendre symbol of a and p where a is an integer and p is an odd prime number.
$[n]P$	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 curve E, where $Q = [n]P = P + P + \ldots + P$, i.e., the sum of n copies of P. The operation satisfies $[0]P = O_E$ and $[-n]P = [n](-P)$.
$[x, y]$	The set of integers from x to y inclusive, if x, y are integers satisfying $x \leq y$.
||	$X \|\| Y$ is used to mean the result of the concatenation of data items X and Y in the order specified. In cases where the result of concatenating two or more data items is signed as part of one of the mechanisms specified in this part of ISO/IEC 20008, 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]

5 General model and requirements

This clause specifies the general model and requirements for the anonymous digital signature mechanisms specified in this part of ISO/IEC 20008. Some of the contents of this clause are taken from Part 1 of this international standard. In addition, specific requirements applying to mechanisms using a group public key are addressed.

An anonymous digital signature mechanism using a group public key involves a group and a set of group members. Each group shall possess a group membership issuer. There may also be a group membership opener and/or a group signature linker, depending on the mechanism. Multiple entities may function in the role of a group membership opener or a group signature linker. The level of anonymity of the mechanism depends on the anonymity strength (i.e., the size of the group), whether there is an opening capability, whether there is a linking capability, how revocation is done, whether the issuer knows the private keys, and the likelihood of compromise of a private key.

Such an anonymous digital signature mechanism is defined by the specification of the following processes:

— key generation process,

— signature process,

— verification process,

— opening process (if the mechanism supports opening),

— linking process (if the mechanism supports linking), and

— revocation process.

The anonymous digital signature mechanisms using a group public key specified in this part of ISO/IEC 20008 involve a range of types of entity. Some of these entities exist in every mechanism whereas others exist only in some mechanisms. These entities are as follows:

— **Signer**: a signer is an entity that generates a digital signature. In some mechanisms, a signer role is split between two entities. For example, in direct anonymous attestation mechanisms, the signer

role is split between a principal signer with limited computational and storage capability, e.g. a trusted platform module (TPM), and an assistant signer with more computational power but less security tolerance, e.g. an ordinary computer platform (namely the Host with an embedded TPM).

— **Verifier**: a verifier is an entity that verifies a digital signature.

— **Group membership issuer**: a group membership issuer is an entity that issues a group membership credential to a signer. This entity exists in all the mechanisms.

— **Group membership opener**: a group membership opener is an entity who can identify the signer from a signature. This entity exists in some of the mechanisms.

— **Group signature linker**: a group signature linker is an entity that checks whether two signatures have been generated by the same signer with a linking key or a linking base. This entity exists in some of the mechanisms.

In order to use any of the mechanisms specified in this part of ISO/IEC 20008, the following requirements shall be met:

— Each entity involved in an anonymous digital signature mechanism is aware of a common set of group public parameters, which are used to compute a variety of functions in the mechanism.

— Each verifier has access to an authentic copy of the group public key.

— An authentic channel is required between a signer and a group membership issuer during the process of issuing group member signature key. This ensures that the group membership issuer is able to provide the group member signature key only to a legitimate group member.

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

— A robust random bit generator such as one of those specified in ISO/IEC 18031 shall be used.

— A robust prime number generator such as one of those specified in ISO/IEC 18032 shall be used.

— A robust elliptic curve generator such as one of those specified in ISO/IEC 15946-5 shall be used in some mechanisms.

6 Mechanisms with linking capability

6.1 General

This clause specifies four digital signature mechanisms with linking capability.

NOTE 1 In the literature the mechanism of 6.2 is called a list signature scheme, and the mechanisms of 6.3, 6.4 and 6.5 are called DAA schemes. The mechanisms given in 6.2, 6.4 and 6.5 are based on schemes originally specified in [9], [6], and [11], respectively, in which security proofs can also be found. The mechanism in 6.3 is based on a scheme in [3] which is a minor modification of the scheme in [4]; the associated security analysis is given in the full version of [4].

NOTE 2 For certain applications such as attestation, a message to be signed may be hashed and/or concatenated with additional information before being input to the signature process of one of the anonymous digital signature mechanisms specified in this clause.

6.2 Mechanism 1

6.2.1 Symbols

The following symbols apply in the specification of this mechanism.

— l_p, k, l_x, l_e, l_E, l_X, ε: security parameters.

— p', q', e: prime numbers.

— a, a_0, g, h, b, C_1, D, C_2, d', d_1, d_2, t', t_1, t_2, A, f, T_1, T_2, T_3, T_4, d_3, d_4, d_5, t_3, t_4, t_5: integers in QR(n).

— x', α, β: integers in $[0, 2^{lx} - 1]$.

— w_1, w_2, w_3: integers in $[0, 2^{2lp} - 1]$.

— $\hat{c}$, $\dot{c}$, c', c, c'', c''': k-bit integers.

— $\check{r}$: $(2l_p + 1)$-bit integer.

— t_1, $\hat{s}_1$, r', r_1, r_2: $(\varepsilon \cdot (l_x + k))$-bit integers.

— t_2, $\hat{s}_2$: $(\varepsilon \cdot (2l_p + k + 1))$-bit integers.

— x: $(l_x + 1)$-bit integer.

— r_3: $(\varepsilon \cdot (l_x + 2l_p + k + 1))$-bit integer.

— s_0, s_1, s_2, s': integers in $[-2^{lx+k}, 2^{\varepsilon(lx+k)} - 1]$.

— s_3: integer in $[-2^{lx+2lp+k+1}, 2^{\varepsilon(lx+2lp+k+1)} - 1]$.

— r_4, r_5: $(\varepsilon \cdot (2l_p + k))$-bit integers.

— r_9, r_{10}: $(\varepsilon \cdot (2l_p + l_e + k))$-bit integers.

— s_4, s_5: integers in $[-2^{2lp+k}, 2^{\varepsilon(2lp+k)} - 1]$.

— s_9, s_{10}: integers in $[-2^{2lp+le+k}, 2^{\varepsilon(2lp+le+k)} - 1]$.

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

— H_I: a hash function that outputs $(2l_p)$-bit message digest.

6.2.2 Key generation process

The key generation process has two parts: a setup process and a group membership issuing process. The setup process is executed by the group membership issuer to create the group public parameter, group public key, and group membership issuing key. The group membership issuing process is an interactive protocol running between the group membership issuer and a group member to create a unique group member signature key for the group member.

The setup process takes the following steps by the group membership issuer:

a) Choose the following parameters: l_p, k, l_x, l_e, l_E, l_X, ε.

b) Choose an RSA modulus $n = pq$ with $p = 2p' + 1$, $q = 2q' + 1$ such that p, q, p', q' are all primes and p' as well as q' have l_p bits.

c) Choose a random generator a of the group of quadratic residues modulo n by performing the following steps:

   1) Choose a random integer g in Z_n^* such that $\gcd(g+1, n) = 1$ and $\gcd(g - 1, n) = 1$.

   2) Compute $a = g^2 \pmod{n}$.

d) Choose a random generator a_0 of QR(n) different from a.

e) Choose a random generator g of QR(n) different from a and a_0.

f) Choose a random generator h of QR(n) different from a, a_0 and g.

g) Choose a random generator b of QR(n) different from a, a_0, g and h.

h) The group membership issuer chooses two hash functions H: $\{0, 1\}^* \rightarrow \{0, 1\}^k$ and H_Γ: $\{0, 1\}^* \rightarrow \{0, 1\}^{2lp}$. An example of how to construct H_Γ is provided in Annex B.

i) Output the following:

— group public parameter = (l_p, k, l_x, l_e, l_E, l_X, ε),

— group public key = (n, a, a_0, g, h, b),

— group membership issuing key = (p', q').

NOTE An example of recommended parameters is provided in Annex C.2.

The group membership issuing process may require a secure and authentic channel between the member and the group membership issuer to prevent the group membership credential from being observed by an eavesdropper. How to establish such a channel is out scope of this mechanism. The group membership issuing process is as follows:

a) The group member chooses a random integer $x' \in [0, 2^{lx} -1]$.

b) The member chooses a random integer $\check{r} \in [0, 2n -1]$.

c) The member computes $C_1 = g^{x'}h^{\check{r}} \pmod n$.

d) The member generates a proof of knowledge U of the representation (x', $\check{r}$) of C_1 in the bases g and h by performing the following steps:

1) The member chooses a random integer $t_1 \in [0, 2^{\varepsilon(lx+k)}-1]$.

2) The member chooses a random integer $t_2 \in [0, 2^{\varepsilon(2lp+k+1)}-1]$.

3) The member computes $D = g^{t1}h^{t2} \pmod n$.

4) The member computes $\hat{c} = H(g \,||\, h \,||\, C_1 \,||\, D)$.

5) The member computes $\hat{s}_1 = t_1 - \hat{c}x'$.

6) The member computes $\hat{s}_2 = t_2 - \hat{c}\,\check{r}$.

7) $U = (\hat{c}, \hat{s}_1, \hat{s}_2)$.

e) The member sends C_1 and U to the group membership issuer.

f) The group membership issuer receives C_1 and U from the member.

g) The group membership issuer verifies that C_1 belongs to QR(n) by performing the following step:

1) The group membership issuer checks that $(C_1|p) = 1$ and that $(C_1|q) = 1$. If either of these verifications fails, the group membership issuer outputs Reject and stops.

h) The group membership issuer verifies the proof of knowledge U by performing the following steps:

1) The group membership issuer computes $D' = g^{\hat{s}1}h^{\hat{s}2}C_1^{\hat{c}} \pmod n$.

2) The group membership issuer computes $\dot{c} = H(g \,||\, h \,||\, C_1 \,||\, D')$.

3) The group membership issuer checks that $\dot{c} = \hat{c}$, $\hat{s}_1$ belongs to $[-2^{lx+k}, 2^{\varepsilon(lx+k)}-1]$ and $\hat{s}_2$ belongs to $[-2^{2lp+k+1}, 2^{\varepsilon(2lp+k+1)}-1]$. If any of these verifications fails, the group membership issuer outputs Reject and stops.

i) The group membership issuer chooses a random odd integer $\alpha \in [0, 2^{lx} -1]$.

j) The group membership issuer chooses a random integer $\beta \in [0, 2^{lx} -1]$.

k) The group membership issuer sends α and β to the member.

l) The member receives α and β from the group membership issuer.

m) The member computes $x = 2^{lX} + (\alpha x' + \beta \pmod{2^{lx}})$.

n) The member computes $C_2 = a^x \pmod{n}$.

o) The member computes $v = (\alpha x' + \beta) \mid 2^{lx}$.

p) The member generates a proof of knowledge V of the discrete logarithm x of C_2 in base a by performing the following steps:

1) The member chooses a random integer $r' \in [0, 2^{\varepsilon(lx+k)}-1]$.

2) The member computes $d' = a^{r'} \pmod{n}$.

3) The member computes $c' = H(a \,||\, g \,||\, C_2 \,||\, d')$.

4) The member computes $s' = r' - c'(x - 2^{lX})$.

5) The member set $V = (c', s')$.

q) The member generates a proof of knowledge W by performing the following steps:

1) The member chooses a random integer $r_1 \in [0, 2^{\varepsilon(lx+k)}-1]$.

2) The member chooses a random integer $r_2 \in [0, 2^{\varepsilon(lx+k)}-1]$.

3) The member chooses a random integer $r_3 \in [0, 2^{\varepsilon(lx+2lp+k+1)}-1]$.

4) The member computes $d_1 = a^{r1} \pmod{n}$.

5) The member computes $d_2 = g^{r1}(g^l)^{r2}h^{r3} \pmod{n}$ where $l = 2^{lx}$.

6) The member computes $c = H(a \,||\, g \,||\, h \,||\, C_1 \,||\, C_2 \,||\, d_1 \,||\, d_2)$.

7) The member computes $s_1 = r_1 - c(x - 2^{lX})$.

8) The member computes $s_2 = r_2 - cv$.

9) The member computes $s_3 = r_3 - c\alpha\check{r}$.

10) The member sets $W = (c, s_1, s_2, s_3)$.

r) The member sends C_2, V and W to the group membership issuer.

s) The group membership issuer receives C_2, V and W from the member.

t) The group membership issuer checks that C_2 belongs to QR(n) by performing the following step:

1) The group membership issuer checks that $(C_2|p) = 1$ and that $(C_2|q) = 1$. If any of these verifications fails, the group membership issuer outputs Reject and stops.

u) The group membership issuer verifies the proof of knowledge V by performing the following steps:

1) The group membership issuer computes $s_0 = s' - c' 2^{lX}$.

2) The group membership issuer computes $t' = C_2{}^{c'}a^{s0} \pmod{n}$.

3) The group membership issuer computes $c'' = H(a \,||\, g \,||\, C_2 \,||\, t')$.

4) The group membership issuer checks that $c'' = c'$ and that $s' \in [-2^{lx+k}, 2^{\varepsilon(lx+k)}-1]$. If any of these verifications fails, the group membership issuer outputs Reject and stops.

v) The group membership issuer verifies the proof of knowledge W by performing the following steps:

1) The group membership issuer computes $t_1 = (C_2/a^L)^c a^{s1} \pmod{n}$ where $L = 2^{lX}$.

2) The group membership issuer computes $t_2 = (C_1{}^{\alpha} g^{\beta})^c g^{s1} (g^l)^{s2} h^{s3} \pmod{n}$ where $l = 2^{lx}$.

3) The group membership issuer computes $c''' = H(a \,||\, g \,||\, h \,||\, C_1 \,||\, C_2 \,||\, t_1 \,||\, t_2)$.

4) The group membership issuer checks that: $c''' = c$, s_1 belongs to $[-2^{lx+k}, 2^{\varepsilon(lx+k)}-1]$, s_2 belongs to $[-2^{lx+k}, 2^{\varepsilon(lx+k)}-1]$ and that s_3 belongs to $[-2^{lx+2lp+k+1}, 2^{\varepsilon(lx+2lp+k+1)}-1]$. If any of these verifications fails, the group membership issuer outputs Reject and stops.

w) The group membership issuer chooses a random prime $e \in [2^{lE} - 2^{le} + 1, 2^{lE} + 2^{le} - 1]$.

x) The group membership issuer computes $đ_1 = 1/e \pmod{p'q'}$.

y) The group membership issuer computes $A = (a_0 C_2)^{đ1} \pmod{n}$.

z) The group membership issuer stores $(A, e, Member)$ in member-list LIST.

aa) The group membership issuer sends A and e to the member.

bb) The member receives A and e from the group membership issuer.

cc) The member checks that $A^e = a_0 a^x \pmod{n}$.

dd) The group member signature key of the signer is (A, e, x), in which x is the group member private key and (A, e) is the group membership credential.

6.2.3 Signature process

On input of a group public key (n, a, a_0, g, h, b), a group member signature key(A, e, x), a linking base bsn, and a message $m \in \{0, 1\}^*$ to be signed, the signature process takes the following steps below. The linking base is used for the linking capability. It is chosen by group membership issuer or any other trusted authorities.

a) The member computes $f = (H_\Gamma(bsn))^2 \pmod{n}$.

b) The member chooses a random integer $w_1 \in [0, 2^{2lp} - 1]$.

c) The member chooses a random integer $w_2 \in [0, 2^{2lp} - 1]$.

d) The member chooses a random integer $w_3 \in [0, 2^{2lp} - 1]$.

e) The member computes $T_1 = Ab^{w1} \pmod{n}$.

f) The member computes $T_2 = g^{w1} h^{w2} \pmod{n}$.

g) The member computes $T_3 = g^e h^{w3} \pmod{n}$.

h) The member computes $T_4 = f^x \pmod{n}$.

i) The member chooses a random integer $r_1 \in [0, 2^{\varepsilon(le+k)}-1]$.

j) The member chooses a random integer $r_2 \in [0, 2^{\varepsilon(lx+k)}-1]$.

k) The member chooses a random integer $r_3 \in [0, 2^{\varepsilon(2lp+k)}-1]$.

l) The member chooses a random integer $r_4 \in [0, 2^{\varepsilon(2lp+k)}-1]$.

m) The member chooses a random integer $r_5 \in [0, 2^{\varepsilon(2lp+k)}-1]$.

n) The member chooses a random integer $r_9 \in [0, 2^{\varepsilon(2lp+le+k)}-1]$.

o) The member chooses a random integer $r_{10} \in [0, 2^{\varepsilon(2lp+le+k)}-1]$.

p) The member computes $d_1 = T_1^{r1}/(a^{r2}b^{r9}) \pmod{n}$.

q) The member computes $d_2 = T_2^{r1}/(g^{r9}h^{r10}) \pmod{n}$.

r) The member computes $d_3 = g^{r3}h^{r4} \pmod{n}$.

s) The member computes $d_4 = g^{r1}h^{r5} \pmod{n}$.

t) The member computes $d_5 = f^{r2} \pmod{n}$.

u) The member computes $c = H(a \| a_0 \| g \| h \| T_1 \| T_2 \| T_3 \| T_4 \| d_1 \| d_2 \| d_3 \| d_4 \| d_5 \| m)$.

v) The member computes $s_1 = r_1 - c(e - 2^{lE})$.

w) The member computes $s_2 = r_2 - c(x - 2^{lX})$.

x) The member computes $s_3 = r_3 - cw_1$.

y) The member computes $s_4 = r_4 - cw_2$.

z) The member computes $s_5 = r_5 - cw_3$.

aa) The member computes $s_9 = r_9 - cew_1$.

bb) The member computes $s_{10} = r_{10} - cew_2$.

cc) The member sets the signature as $\sigma = (c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$.

6.2.4 Verification process

On input of a message m, a linking base bsn, a signature $(c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$, a group public key (n, a, a_0, g, h, b), the verification process takes the following steps:

a) The verifier computes $f = (H_f(bsn))^2 \pmod{n}$.

b) The verifier computes $t_1 = a_0^{c}T_1^{s1-cl'} / (a^{s2-cL}b^{s9}) \pmod{n}$ where $l' = 2^{lE}$ and $L = 2^{lX}$.

c) The verifier computes $t_2 = T_2^{s1-cl'} / (g^{s9}h^{s10}) \pmod{n}$ where $l' = 2^{lE}$.

d) The verifier computes $t_3 = T_2^{c}g^{s3}h^{s4} \pmod{n}$.

e) The verifier computes $t_4 = T_3^{c}g^{s1-cl'}h^{s5} \pmod{n}$ where $l' = 2^{lE}$.

f) The verifier computes $t_5 = T_4^{c}f^{s2-cL} \pmod{n}$ where $L = 2^{lX}$.

g) The verifier computes $c' = H(a \| a_0 \| g \| h \| T_1 \| T_2 \| T_3 \| T_4 \| t_1 \| t_2 \| t_3 \| t_4 \| t_5 \| m)$.

h) If $c' = c$, s_1 belongs to $[-2^{le+k}, 2^{\varepsilon(le+k)}-1]$, s_2 belongs to $[-2^{lx+k}, 2^{\varepsilon(lx+k)}-1]$, s_3 belongs to $[-2^{2lp+k}, 2^{\varepsilon(2lp+k)}-1]$, s_4 belongs to $[-2^{2lp+k}, 2^{\varepsilon(2lp+k)}-1]$, s_5 belongs to $[-2^{2lp+k}, 2^{\varepsilon(2lp+k)}-1]$, s_9 belongs to $[-2^{2lp+le+k}, 2^{\varepsilon(2lp+le+k)}-1]$, s_{10} belongs to $[-2^{2lp+le+k}, 2^{\varepsilon(2lp+le+k)}-1]$, then return 1(valid).

i) Else return 0 (invalid).

6.2.5 Linking process

Given two valid signatures $\sigma = (c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$ and $\sigma' = (c', s_1', s_2', s_3', s_4', s_5', s_9', s_{10}', T_1', T_2', T_3', T_4')$ computed using a linking base bsn, the linking process takes the following step:

a) If $T_4 = T_4'$, output 1 (linked), otherwise, output 0 (not linked).

6.2.6 Revocation process

Details of the revocation process in this mechanism are surveyed in [10]. There are two types of revocation (private key revocation and verifier blacklist revocation) supported in this mechanism. Private key revocation can be either a global revocation or a local revocation. Verifier blacklist revocation is a local revocation.

Private key revocation:

— If a group member signature key (A, e, x) is compromised, the group membership issuer or a verifier puts x into a revocation list RL of this type.

— Given a valid signature $\sigma = (c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$ computed using a linking base *bsn* and a revocation list RL of this type, a verifier can check revocation of this signature as follows: For each $x' \in$ RL, verify $T_4 \neq (H_\Gamma(bsn))^{2x'} \pmod{n}$. If any of the verification fails, output 0 (revoked), otherwise, output 1 (valid).

NOTE The private key revocation works only if the group membership issuer or the verifier has learned the group member signature keys of the compromised group members.

Verifier blacklist revocation:

— If signatures were computed using a linking base *bsn*, and a verifier can build its own revocation list RL corresponding to *bsn*. If the verifier wants to blacklist the signer of a signature $\sigma = (c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$, she puts T_4 into a revocation list RL of this type.

— Given a valid signature $\sigma = (c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)$ computed using a linking base *bsn* and a revocation list RL of this type, a verifier can check revocation of this signature as follows: For each $T_4' \in$ RL, verify $T_4 \neq T_4'$. If any of the verification fails, output 0 (revoked), otherwise, output 1 (valid).

NOTE In order to use verifier blacklist revocation in this mechanism, a signer is required to use a specific linking base for each verifier. The value of the linking base could, for example, be chosen by the verifier or agreed in advance by the signer and verifier.

6.3 Mechanism 2

6.3.1 Symbols

The following symbols apply in the specification of this mechanism.

— $l_n, l_f, l_e, l'_e, l_v, l_\emptyset, l_H, l_r, l_s, l_\Gamma, l_\rho$: security parameters.

— p', q', ρ, Γ, e: prime numbers.

— $g', g, h, S, Z, R_0, R_L, U, U', A, A', T, T_t'$: integers in QR($n$).

— $x_0, x_1, x_Z, x_s, x_h, x_g, s_e, r_e$: integers in $[1, p' \cdot q']$.

— $\gamma, J_L, K_L, K_I', J, K, J', K'$: integers whose multiplicative order modulo Γ is ρ.

— f, f': integers in $[0, \rho-1]$.

— f_0, f_1: l_f-bit integers.

— c_h, c, c', n_I, n_V: l_H-bit integers.

— n_T, n_H: $l_\emptyset$-bit integers.

— t, t_2, r_f: $(2l_f + l_\emptyset + l_H + 1)$-bit integers.

— t_1: $(l_e + l_H)$-bit integer.

— r_0, r_1: $(l_f + l_\emptyset + l_H)$-bit integers.

— r_v: $(l_v + l_\emptyset + l_H)$-bit integer.

— r_{v^*}: $(l_e + l_n + 2l_\emptyset + l_H + 1)$-bit integer.

— $r_{v'}$: $(l_n + 2l_\emptyset + l_H)$-bit integer.

— $s_{v'}$: $(l_n + 2l_\emptyset + l_H + 1)$-bit integer.

— s_0, s_1: $(l_f + l_\emptyset + l_H + 1)$-bit integers.

— v', w: $(l_n + l_\emptyset)$-bit integers.

— v^*: $(l_v - 1)$-bit integer.

— v'': l_v-bit integer.

— v: $(l_n + l_\emptyset + l_v + 1)$-bit integer.

— bsn_I: a linking base of the group membership issuer.

— H_Γ: a hash function that outputs $(l_\Gamma + l_\emptyset)$-bit message digest.

6.3.2 Key generation process

The key generation process has two parts: a setup process and a group membership issuing process. The setup process is executed by the group membership issuer to create the group public parameter, group public key, and group membership issuing key. The group membership issuing process is an interactive protocol running between the group membership issuer and a group member to create a unique group member signature key for the group member. Revocation check during the group membership issuing process is required to prevent an attacker from re-enrolling a revoked group member private key.

The setup process takes the following steps by the group membership issuer:

a) Choose the following parameters: l_n, l_f, l_e, l'_e, l_v, $l_\emptyset$, l_H, l_r, l_s, l_Γ, l_ρ.

b) Choose two hash functions H: $\{0, 1\}^* \rightarrow \{0, 1\}^{lH}$ and H_Γ: $\{0, 1\}^* \rightarrow \{0, 1\}^{\ l\Gamma + l\emptyset}$. An example of how to construct H_Γ is provided in Annex B.

c) Choose an RSA modulus $n = pq$ with $p = 2p' + 1$, $q = 2q' + 1$ such that p, q, p', q' are all primes and n has l_n bits.

d) Choose a random integer g' in QR(n).

e) Choose random integers x_0, x_1, x_z, x_s, x_h, x_g in $[1, p'{\cdot}q']$.

f) Compute $g = (g')^{xg} \bmod n$, $h = (g')^{xh} \bmod n$, $S = h^{xs} \bmod n$.

g) Compute $Z = h^{xz} \bmod n$, $R_0 = S^{x0} \bmod n$, $R_1 = S^{x1} \bmod n$.

h) Choose a random prime ρ of l_ρ bits.

i) Choose a random prime Γ of l_Γ bits such that $\Gamma - 1$ is a multiple of ρ and $(\Gamma - 1)/\rho$ is not a multiple of ρ.

j) Choose a random γ, a number whose multiplicative order modulo Γ is ρ.

k) Output the following:

— group public parameter = $(l_n, l_f, l_e, l'_e, l_v, l_\emptyset, l_H, l_r, l_s, l_\Gamma, l_\rho, H, H_\Gamma)$,

— group public key = $(n, g', g, h, S, Z, R_0, R_1, \gamma, \Gamma, \rho)$,

— group membership issuing key = (p', q').

NOTE An example of recommended parameters is provided in Annex C.2.

The group membership issuing process requires a secure and authentic channel between the principal signer and the group membership issuer. How to establish such a channel is out scope of this mechanism. The group membership issuing process includes the following steps:

a) The principal signer and the group membership issuer agree on a linking base bsn_I.

b) The assistant signer computes $J_I = (H_\Gamma(1 \,||\, bsn_I))^{(\Gamma-1)/\rho} \bmod \Gamma$ and sends J_I to the principal signer.

NOTE As discussed in [19], holding the signer anonymity requires that the value bsn_I is never used as bsn in any DAA signatures. In an application where this requirement is not guaranteed, changing $J_I = (H_\Gamma(1 \,||\, bsn_I))^{(\Gamma-1)/\rho} \bmod \Gamma$ to $J_I = (H_\Gamma(0 \,||\, bsn_I))^{(\Gamma-1)/\rho} \bmod \Gamma$ can avoid this potential problem.

c) The principal signer checks that $(J_I)^\rho \equiv 1 \pmod{\Gamma}$.

d) The principal signer chooses a random $f \in [0, \rho-1]$ or derives f from its secret seed value.

e) The principal signer computes $f_1 = \left\lfloor f/2^{lf} \right\rfloor$.

f) The principal signer computes $f_0 = f - 2^{lf} \cdot f_1$. The (f_0, f_1) pair is part of the group member private key.

g) The principal signer randomly picks an integer v' of $(l_n + l_\emptyset)$-bit.

h) The principal signer computes $U = (R_0^{f0} \cdot R_1^{f1} \cdot S^{v'}) \bmod n$.

i) The principal signer computes $K_I = (J_I)^f \bmod \Gamma$.

j) The principal signer sends U and K_I to the assistant signer who forwards them to the group membership issuer.

NOTE After the above step, the group membership issuer can compute $f = f_0 + f_1 \cdot 2^{lf}$ for each (f_0, f_1) on the revocation list and verify $K_I \neq J_I{}^f \bmod \Gamma$. If the signer has been revoked, the group membership issuer aborts the group membership issuing process.

k) The principal signer randomly picks two integers r_0, r_1 of $(l_f + l_\emptyset + l_H)$-bit.

l) The principal signer randomly picks an integer $r_{v'}$ of $(l_n + 2l_\emptyset + l_H)$-bit.

m) The principal signer computes $U' = (R_0^{r0} \cdot R_1^{r1} \cdot S^{rv'}) \bmod n$.

n) The principal signer computes $r_f = r_0 + r_1 \cdot 2^{lf}$.

o) The principal signer computes $K_I' = (J_I)^{rf} \bmod \Gamma$.

p) The signer sends U' and K_I' to the assistant signer.

q) The group membership issuer chooses a random $n_I \in \{0, 1\}^{lH}$ and sends n_I to the assistant signer.

r) The assistant signer computes $c_h = H(n \,||\, R_0 \,||\, R_1 \,||\, S \,||\, U \,||\, K_I \,||\, U' \,||\, K_I' \,||\, n_I)$.

s) The assistant signer sends c_h to the principal signer.

t) The principal signer chooses a random nonce $n_T \in \{0, 1\}^{l\emptyset}$.

u) The principal signer computes $c = H(c_h \,||\, n_T)$.

v) The principal signer computes $s_{v'} = r_{v'} + c \cdot v'$, $s_0 = r_0 + c \cdot f_0$, $s_1 = r_1 + c \cdot f_1$.

w) The principal signer sends $(c, n_T, s_0, s_1, s_{v'})$ to the assistant signer.

x) The assistant signer forwards $(c, n_T, s_0, s_1, s_{v'})$ to the group membership issuer.

y) The group membership issuer verifies that s_0 and s_1 are at most $(l_f + l_\emptyset + l_H + 1)$-bit integers.

z) The group membership issuer verifies that $s_{v'}$ is an at most $(l_n + 2l_\emptyset + l_H + 1)$-bit integer.

aa) The group membership issuer computes $U' = (U^{-c} \cdot R_0^{s0} \cdot R_1^{s1} \cdot S^{sv'}) \bmod n$.

bb) The group membership issuer computes $t = s_0 + s_1 \cdot 2^{lf}$.

cc) The group membership issuer computes $K_I' = (K_I^{-c} \cdot J_I^{t}) \bmod \Gamma$.

dd) The group membership issuer verifies that $c = H(H(n \,||\, R_0 \,||\, R_1 \,||\, S \,||\, U \,||\, K_I \,||\, U' \,||\, K_I' \,||\, n_I) \,||\, n_T)$.

ee) The group membership issuer randomly chooses v^* of $(l_v - 1)$-bit.

ff) The group membership issuer randomly choose a prime e from $[2^{le-1}, 2^{le-1} + 2^{l'e-1}]$.

gg) The group membership issuer computes $v'' = v^* + 2^{lv-1}$.

hh) The group membership issuer computes $A = (Z \cdot U^{-1} \cdot S^{-v''})^{1/e} \bmod n$.

ii) The assistant signer chooses a random integer $n_H \in \{0, 1\}^{l\emptyset}$ and sends to the group membership issuer.

jj) The group membership issuer randomly chooses r_e from $[0, p' \cdot q']$.

kk) The group membership issuer computes $A' = (Z \cdot U^{-1} \cdot S^{-v''})^{re} \bmod n$.

ll) The group membership issuer computes $c' = H(n \,||\, Z \,||\, S \,||\, U \,||\, v'' \,||\, A \,||\, A' \,||\, n_H)$.

mm) The group membership issuer computes $s_e = (r_e - c'/e) \bmod p' \cdot q'$.

nn) The group membership issuer sends c', s_e, and (A, e, v'') to the assistant signer.

oo) The assistant signer verifies that e is a prime in $[2^{le-1}, 2^{le-1} + 2^{l'e} - 1]$.

pp) The assistant signer computes $A' = (A^{c'} \cdot (Z \cdot U^{-1} \cdot S^{-v''})^{se}) \bmod n$.

qq) The assistant signer verifies that $c' = H(n \,||\, Z \,||\, S \,||\, U \,||\, v'' \,||\, A \,||\, A' \,||\, n_H)$.

rr) The assistant signer forward v'' to the principal signer.

ss) The principal signer sets $v = v' + v''$ and stores (f_0, f_1, v) while the assistant signer stores (A, e).

tt) The group member signature key of the signer is (f_0, f_1, A, e, v), in which (f_0, f_1, v) is the group member private key and (A, e) is the group membership credential.

6.3.3 Signature process

On input of a group public key $(n, g', g, h, S, Z, R_0, R_1, \gamma, \Gamma, \rho)$, a group member signature key (f_0, f_1, A, e, v), a linking base bsn, a nonce $n_V \in \{0, 1\}^{lH}$, and a message $m \in \{0, 1\}^*$ to be signed, the signature process takes the following steps below. The linking base is either a special symbol $\perp$ or an arbitrary string used for the linking capability. It is chosen either by the signer or the verifier, or pre-negotiated by both them.

NOTE 1 The nonce n_V is usually chosen by the verifier.

NOTE 2 Alternative way to handle n_V is to include n_V as part of the message m.

a) The principal signer has the group member private key (f_0, f_1, v) while the assistant signer has (A, e).

b) If $bsn = \perp$, the assistant signer chooses a random integer t from $[0, \rho - 1]$ and computes $J = (\gamma)^t \bmod \Gamma$.

c) If $bsn \neq \perp$, the assistant signer computes $J = (H_\Gamma(1 \,||\, bsn))^{(\Gamma-1)/\rho} \bmod \Gamma$.

d) The assistant signer sends J to the principal signer.

e) The principal signer verifies that $(J)^\rho \equiv 1 \bmod \Gamma$.

f) The principal signer computes $f = f_0 + f_1 \cdot 2^{lf}$.

g) The principal signer computes $K = (J)^f \bmod \Gamma$.

h) The principal signer randomly picks an integer r_v of $(l_v + l_\emptyset + l_H)$-bit.

i) The principal signer randomly picks two integers r_0, r_1 of $(l_f + l_\emptyset + l_H)$-bit.

j) The principal signer computes $T_t' = (R_0^{r0} \cdot R_1^{r1} \cdot S^{rv}) \bmod n$.

k) The principal signer computes $r_f' = (r_0 + r_1 \cdot 2^{lf}) \bmod \rho$.

l) The principal signer computes $K' = (J)^{rf'} \bmod \Gamma$.

m) The principal signer sends K, T_t', K' to the assistant signer.

n) The assistant signer randomly picks an integer w of $(l_n + l_\emptyset)$-bit.

o) The assistant signer randomly picks an integer r_e of $(l'_e + l_\emptyset + l_H)$-bit.

p) The assistant signer randomly picks an integer r_{v^*} of $(l_e + l_n + 2l_\emptyset + l_H + 1)$-bit.

q) The assistant signer computes $T = (A \cdot S^w) \bmod n$.

r) The assistant signer computes $T' = (T_t' \cdot T^{re} \cdot S^{rv'}) \bmod n$.

s) The assistant signer computes $c_h = H(n \,||\, R_0 \,||\, R_1 \,||\, S \,||\, Z \,||\, \gamma \,||\, \Gamma \,||\, \rho \,||\, J \,||\, T \,||\, K \,||\, T' \,||\, K' \,||\, n_V)$.

t) The assistant signer sends c_h to the principal signer.

u) The principal signer chooses a random nonce $n_T \in \{0, 1\}^{l\emptyset}$.

v) The principal signer computes $c = H(H(c_h \,||\, n_T) \,||\, m)$.

w) The principal signer computes $s_v = r_v + c \cdot v$, $s_0 = r_0 + c \cdot f_0$, $s_1 = r_1 + c \cdot f_1$.

x) The principal signer sends (c, n_T, s_v, s_0, s_1) to the assistant signer.

y) The assistant signer computes $s_e = r_e + c \cdot (e - 2^{le-1})$.

z) The assistant signer computes $s_{v'} = s_v + r_{v^*} - c \cdot w \cdot e$.

aa) The assistant signer outputs the group signature $\sigma = (J, K, T, c, n_T, s_{v'}, s_0, s_1, s_e)$.

6.3.4 Verification process

On input of a message m, a linking base bsn, a nonce $n_V \in \{0, 1\}^{lH}$, a group signature $(J, K, T, c, n_T, s_{v'}, s_0, s_1, s_e)$, a group public key $(n, g', g, h, S, Z, R_0, R_1, \gamma, \Gamma, \rho)$, the verification process takes the following steps:

a) Verify that $(J)^\rho \equiv 1 \bmod \Gamma$ and $(K)^\rho \equiv 1 \bmod \Gamma$.

b) Verify that s_0 and s_1 are at most $(l_f + l_\emptyset + l_H + 1)$-bit integers.

c) Verify that s_e is an at most $(l'_e + l_\emptyset + l_H + 1)$-bit integer.

d) Compute $t_1 = s_e + c \cdot 2^{le-1}$.

e) Compute $t_2 = s_0 + s_1 \cdot 2^{lf}$.

f) Compute $T' = (Z^{-c} \cdot T^{t1} \cdot R_0^{s0} \cdot R_1^{s1} \cdot S^{sv'}) \bmod n$.

g) Compute $K' = (K^{-c} \cdot J^{t2}) \bmod \Gamma$.

h) Verify that $c = H(H(H(n \,||\, R_0 \,||\, R_1 \,||\, S \,||\, Z \,||\, \gamma \,||\, \Gamma \,||\, \rho \,||\, J \,||\, T \,||\, K \,||\, T' \,||\, K' \,||\, n_V) \,||\, n_T) \,||\, m)$.

i) If J is derived from linking base bsn, verify that $J = (H_\Gamma(1 \,||\, bsn))^{(\Gamma-1)/\rho} \bmod \Gamma$.

j) Optionally, call the revocation checking process.

k) If any of the above verifications fails, output 0 (invalid), otherwise, output 1 (valid).

NOTE The revocation check in the verification process could be performed in the first step instead of the last step of the process.

6.3.5 Linking process

Given two signatures $\sigma = (J, K, T, c, n_T, s_{v'}, s_0, s_1, s_e)$ and $\sigma' = (J', K', T', c', n_T', s_{v'}', s_0', s_1', s_e')$, the linking process takes the following step:

a) If $J = J'$ and $K = K'$, output 1 (linked), otherwise, output 0 (not linked).

NOTE If the linking process outputs 0 because of $J \neq J'$, it means that the linking process cannot determine whether two signatures were created by the same group member.

6.3.6 Revocation process

Details of the revocation process in this mechanism are surveyed in [10]. There are two types of revocation (private key revocation and verifier blacklist revocation) supported in this mechanism. Private key revocation can be either a global revocation or a local revocation. Verifier blacklist revocation is a local revocation.

Private key revocation:

— If a group member signature key (f_0, f_1, A, e, v) is compromised, the group membership issuer or a verifier computes $f = f_0 + f_1 \cdot 2^{lf}$, and puts f into a revocation list RL of this type.

— Given a signature $\sigma = (J, K, T, c, n_T, s_{v'}, s_0, s_1, s_e)$ and a revocation list RL of this type, a verifier can check revocation of this signature as follows: For each $f' \in$ RL, verify $K \neq (J)^{f'} \bmod \Gamma$. If any of the verification fails, output 0 (revoked), otherwise, output 1 (valid).

NOTE The private key revocation works only if the group membership issuer or the verifier has learned the group member signature keys of the compromised group members.

Verifier blacklist revocation:

— If signatures were computed using a linking base bsn, and a verifier can build its own revocation list RL corresponding to bsn. If the verifier wants to blacklist the signer of a signature $\sigma = (J, K, T, c, n_T, s_{v'}, s_0, s_1, s_e)$, she put K into revocation RL of this type.

— Given a signature $\sigma = (J, K, T, c, n_T, s_{v'}, s_0, s_1, s_e)$ and a revocation list RL of this type, a verifier can check revocation of this signature as follows: For each $K' \in$ RL, verify $K \neq K'$. If any of the verification fails, output 0 (revoked), otherwise, output 1 (valid).

NOTE In order to use verifier blacklist revocation in this mechanism, a signer is required to use a specific linking base for each verifier. The value of the linking base could, for example, be chosen by the verifier or agreed in advance by the signer and verifier.

6.4 Mechanism 3

6.4.1 Symbols

The following symbols apply in the specification of this mechanism.

— t: a security parameter.

— $Q_1, Q_2, A, F, R, J, K, J', K', T, R_1, R_{2t}, R_3$: elements of G_1.

— W: elements of G_2.

— T_1, T_2, T_3, T_4, R_2: elements of G_T.

— $y, f, f', x, r, c, c_h, a, b, r_f, r_x, r_a, r_b, s_f, s_x, s_a, s_b, u, v, r_u, r_v, s_u, s_v$: integers in Z_p.

— n_I, n_V, n_T: t-bit integers.

— H_1: a hash function that outputs elements in Z_p.

— H_2: a hash functions that output elements in G_1.

6.4.2 Key generation process

The key generation process takes the following steps by the group membership issuer:

a) Choose an asymmetric bilinear group pair (G_1, G_2) of large prime order p and an associated pairing function e: $G_1 \times G_2 \rightarrow G_T$.

b) Choose a random generator P_1 of G_1.

c) Choose a random generator P_2 of G_2.

d) Choose two hash functions H_1: $\{0, 1\}^* \rightarrow Z_p$ and H_2: $\{0, 1\}^* \rightarrow G_1$. An example of how to construct such hash functions is provided in Annex B.

e) Choose random elements Q_1, Q_2 from G_1.

f) Choose a random integer y from Z_p^* and computes $W = [y]P_2$.

g) Compute $T_1 = e(P_1, P_2)$, $T_2 = e(Q_1, P_2)$, $T_3 = e(Q_2, P_2)$, and $T_4 = e(Q_2, W)$.

h) Output the following:

— group public parameter = (G_1, G_2, G_T, p, e, P_1, P_2, H_1, H_2),

— group public key = (Q_1, Q_2, W, T_1, T_2, T_3, T_4),

— group membership issuing key = y.

NOTE 1 T_1, T_2, T_3, and T_4 are optional in group public key, as they can be computed from P_1, P_2, Q_1, Q_2, and W by the signers and verifiers.

NOTE 2 Examples of recommended parameters are provided in Annex C.2.

The group membership issuing process requires a secure and authentic channel between the signer and the group membership issuer. How to establish such a channel is out scope of this mechanism.

For each signer, the group membership issuer generates a group member signature key as follows:

a) Choose two random integers f, x from Z_p^* or derive them from a secret seed value.

b) Compute $A = [1/(x+y)](P_1 + [f]Q_1)$.

c) Output the group member signature key as (f, A, x).

NOTE If the group member signature keys are generated using the above method, this mechanism does not provide unlinkability with respect to the group membership issuer, unless the group membership issuer deletes all the group member signature keys after provisioning them to the signers.

Alternatively, each signer can run a group membership issuing process with the group membership issuer to obtain a group member signature key:

a) The group membership issuer chooses a nonce $n_I \in \{0, 1\}^t$.

b) The group membership issuer sends n_I to the signer.

c) The signer chooses at random a group member private key f from Z_p^* or derives f from its secret seed value.

d) The signer chooses at random integer r from Z_p^*.

e) The signer computes $F = [f]Q_1$ and $R = [r]Q_1$.

f) The signer computes $c = H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, F \,||\, R \,||\, n_I)$.

g) The signer computes $s = (r + c \cdot f) \bmod p$.

h) The signer sends (F, c, s) to the group membership issuer.

i) The group membership issuer computes $R = [s]Q_1 - [c]F$.

j) The group membership issuer verifies that $c = H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, F \,||\, R \,||\, n_I)$.

k) The group membership issuer chooses at random integer x from Z_p^*.

l) The group membership issuer computes $A = [1/(x+y)](P_1 + F)$.

m) The group membership issuer sets (A, x) as the signer's group membership credential and sends it to the signer.

n) The signer verifies the credential by checking that $e(A, W + [x]P_2) = e(P_1 + F, P_2)$.

o) The group member signature key for the signer is (f, A, x).

6.4.3 Signature process

On input of a group public key $(Q_1, Q_2, W, T_1, T_2, T_3, T_4)$, a group member signature key (f, A, x), a linking base bsn, and a message $m \in \{0, 1\}^*$ to be signed, the signature process takes the following steps. The linking base is either a special symbol $\perp$ or an arbitrary string used for the linking capability.

a) If $bsn = \perp$, choose a random element J from G_1, otherwise, compute $J = H_2(bsn)$.

b) Compute $K = [f]J$.

c) Choose a random integer a from Z_p^* and compute $b = (a \cdot x) \bmod p$.

d) Compute $T = A + [a]Q_2$.

e) Randomly pick four random integers r_f, r_x, r_a, r_b from Z_p^*.

f) Compute $R_1 = [r_f]J$.

g) Compute $R_2 = e(A, P_2)^{-rx} \cdot T_2^{rf} \cdot T_3^{rb - a \cdot rx} \cdot T_4^{ra}$.

NOTE $e(A, P_2)$ can be pre-computed by the group member and re-used in each signature process.

h) Compute $c = H_1(H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, J \,||\, K \,||\, T \,||\, R_1 \,||\, R_2) \,||\, m)$.

i) Compute $s_f = (r_f + c \cdot f) \bmod p$, $s_x = (r_x + c \cdot x) \bmod p$.

j) Compute $s_a = (r_a + c \cdot a) \bmod p$, $s_b = (r_b + c \cdot b) \bmod p$.

k) Output the anonymous digital signature $\sigma = (J, K, T, c, s_f, s_x, s_a, s_b)$.

The signature process can be jointly performed by a principal signer and an assistant signer as follows.

a) The principal signer has the group member private key f while the assistant signer has (A, x).

b) If $bsn = \perp$, the principal signer chooses a random element J from G_1, otherwise, computes $J = H_2(bsn)$.

c) The principal signer computes $K = [f]J$.

d) The principal signer chooses a random integer r_f from Z_p^*.

e) The principal signer computes $R_1 = [r_f]J$ and $R_{2t} = [r_f]Q_1$.

f) The principal signer sends (J, K, R_1, R_{2t}) to the assistant signer.

g) The assistant signer chooses a random integer a from Z_p^* and computes $b = (a \cdot x) \bmod p$.

h) The assistant signer computes $T = A + [a]Q_2$.

i) The assistant signer randomly picks three integers r_x, r_a, r_b from Z_p^*.

j) The assistant signer computes $R_2 = e(R_{2t} - [r_x]T + [r_b]Q_2, P_2) \cdot T_4^{r_a}$.

k) The assistant signer computes $c_h = H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, J \,||\, K \,||\, T \,||\, R_1 \,||\, R_2)$.

l) The assistant signer sends c_h to the principal signer.

m) The principal signer chooses a nonce $n_T \in \{0, 1\}^t$.

n) The principal signer computes $c = H_1(c_h \,||\, n_T \,||\, m)$.

o) The principal signer computes $s_f = (r_f + c \cdot f) \bmod p$.

p) The principal signer sends (c, n_T, s_f) to the assistant signer.

q) The assistant signer computes $s_x = (r_x + c \cdot x) \bmod p$, $s_a = (r_a + c \cdot a) \bmod p$, $s_b = (r_b + c \cdot b) \bmod p$.

r) The assistant signer outputs the anonymous digital signature $\sigma = (J, K, T, c, n_T, s_f, s_x, s_a, s_b)$.

NOTE 1 The nonce n_T chosen by the principal signer is optional and can be omitted.

NOTE 2 The signature process may include a verifier's nonce n_V as an optional input. If n_V is included as input, step h) of the signature process computes $c = H_1(H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, J \,||\, K \,||\, T \,||\, R_1 \,||\, R_2 \,||\, n_V) \,||\, m)$ and step k) of the joint signature process computes $c_h = H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, J \,||\, K \,||\, T \,||\, R_1 \,||\, R_2 \,||\, n_V)$.

6.4.4 Verification process

On input of a message m, a linking base bsn, a signature $(J, K, T, c, s_f, s_x, s_a, s_b)$, a group public key $(Q_1, Q_2, W, T_1, T_2, T_3, T_4)$, the verification process takes the following steps:

a) Verify that J, K, T are elements in $\mathcal{G}_1$.

b) Verify that s_f, s_x, s_a, s_b are integers in Z_p.

c) If $bsn \neq \perp$, verify that $J = H_2(bsn)$.

d) Compute $R_1 = [s_f]J - [c]K$.

e) Compute $R_2 = e(T, [-s_x]P_2 - [c]W) \cdot T_1^{c} \cdot T_2^{s_f} \cdot T_3^{s_b} \cdot T_4^{s_a}$.

f) Verify that $c = H_1(H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, J \,||\, K \,||\, T \,||\, R_1 \,||\, R_2) \,||\, m)$.

g) Optionally, call the revocation checking process.

h) If any of the above verifications fails, output 0 (invalid), otherwise, output 1 (valid).

NOTE If nonce n_T is included in the signature, then step f) of the verification process verifies that $c = H_1(H_1(p \,||\, P_1 \,||\, P_2 \,||\, Q_1 \,||\, Q_2 \,||\, W \,||\, J \,||\, K \,||\, T \,||\, R_1 \,||\, R_2) \,||\, n_T \,||\, m)$.

6.4.5 Linking process

Given two signatures $\sigma = (J, K, T, c, n_T, s_f, s_x, s_a, s_b)$ and $\sigma' = (J', K', T', c', n_T', s_f', s_x', s_a', s_b')$, the linking process takes the following step:

a) If $J = J'$ and $K = K'$, output 1 (linked), otherwise, output 0 (not linked).

NOTE If the linking process outputs 0 because of $J \neq J'$, it means that the linking process cannot determine whether two signatures were created by the same group member.

6.4.6 Revocation process

Details of the revocation process in this mechanism are surveyed in [10]. There are three types of revocation (private key revocation, verifier blacklist revocation, and signature revocation) supported in this mechanism. Private key revocation and signature revocation can be either global revocation or local revocation. Verifier blacklist revocation is a local revocation.

Private key revocation:

— If a group member signature key (f, A, x) is compromised, the group membership issuer or a verifier puts f into a revocation list RL of this type.

— Given a signature $\sigma = (J, K, T, c, n_T, s_f, s_x, s_a, s_b)$ and a revocation list RL of this type, a verifier can check revocation of this signature as follows: For each $f' \in$ RL, verify $K \neq [f']J$. If any of the verification fails, output 0 (revoked), otherwise, output 1 (valid).

Verifier blacklist revocation:

— If signatures were computed using a linking base bsn, and a verifier can build its own revocation list RL corresponding to bsn. If the verifier wants to blacklist the signer of a signature $\sigma = (J, K, T, c, n_T, s_f, s_x, s_a, s_b)$, she put K into revocation RL of this type.

— Given a signature $\sigma = (J, K, T, c, n_T, s_f, s_x, s_a, s_b)$ and a revocation list RL of this type, a verifier can check revocation of this signature as follows: For each $K' \in$ RL, verify $K \neq K'$. If any of the verification fails, output 0 (revoked), otherwise, output 1 (valid).

NOTE In order to use verifier blacklist revocation in this mechanism, a signer is required to use a specific linking base for each verifier. The value of the linking base could, for example, be chosen by the verifier or agreed in advance by the signer and verifier.

Signature revocation:

— If the group membership issuer or a verifier determines that a signature $\sigma = (J, K, T, c, n_T, s_f, s_x, s_a, s_b)$ was created by a corrupted signer but has not obtained the corresponding group member signature key, the issuer or the verifier places (J, K) pair of the signature into a revocation list RL of this type.

— In order to perform revocation check, the signer needs to use zero-knowledge proof to prove that she did not create any of the (J', K') in RL before. More specifically, let $\sigma = (J, K, T, c, n_T, s_f, s_x, s_a, s_b)$ be the signature that the signer just created, for each (J', K') pair in RL, the signer proves to the verifier in zero-knowledge that $[f]J = K$ and $[f]J' \neq K'$ as follows:

  a) The signer chooses a random integer u from Z_p^*.

  b) The signer computes $v = (-f \cdot u) \bmod p$.

  c) The signer computes $T = [u]K' + [v]J'$. If $T = O_E$, non-revoked proof fails.

  d) The signer chooses two random integer r_u, r_v from Z_p^*.

  e) The signer computes $R_1 = [r_u]K + [r_v]J$ and $R_3 = [r_u]K' + [r_v]J'$.

  f) The signer computes $c = H_1(p \,||\, P_1 \,||\, J \,||\, K \,||\, J' \,||\, K' \,||\, T \,||\, R_1 \,||\, R_3 \,||\, m)$.

  g) The signer computes $s_u = (r_u + c \cdot u) \bmod p$ and $s_v = (r_v + c \cdot v) \bmod p$.

  h) The signer sends (T, c, s_u, s_v) as the non-revoked proof to the verifier.

  i) The verifier verifies that $T \in G_1$ and $s_u, s_v \in Z_p$.

  j) The verifier verifies that $T \neq O_E$.

k) The verifier computes $R_1 = [s_u]K + [s_v]J$ and $R_3 = [s_u]K' + [s_v]J' - [c]T$.

l) The verifier verifies that $c = H_1(p \,||\, P_1 \,||\, J \,||\, K \,||\, J' \,||\, K' \,||\, T \,||\, R_1 \,||\, R_3 \,||\, m)$.

m) If any of the verification steps fails, the verifier rejects the non-revoked proof, otherwise, accepts the proof.

NOTE 1 The signature revocation requires an interactive process between the signer and verifier.

NOTE 2 To preserve anonymity, it is recommended to have a trusted entity for updating the signature revocation list. If a malicious entity controls the signature revocation list, the anonymity of the signer can be reduced.

6.5 Mechanism 4

6.5.1 Symbols

The following symbols apply in the specification of this mechanism.

— t: a security parameter.

— $Q_2, U', A, B, C, D, R, S, T, W, J, K, R_1, R_2, C'$: elements of G_1.

— X, Y, X': elements of G_2.

— $x, y, f, u, v, w, v', l, c, r, h, s, x', \beta$: integers in Z_p.

— n_I, n_V, n_T: t-bit integers.

— H_1: a hash function that outputs elements in G_1.

— H_2, H_3, H_4: hash functions that output elements in Z_p.

6.5.2 Key generation process

The key generation process has two parts: setup process and group membership issuing process. The setup process is executed by the group membership issuer to create the group public parameter, group public key, and group membership issuing key. The group membership issuing process is an interactive protocol running between the group membership issuer and a group member to create a unique group member signature key for the group member.

The setup process takes the following steps by the group membership issuer:

a) Choose t as a security parameter.

b) Choose an asymmetric bilinear group pair (G_1, G_2) of large prime order p and an associated pairing function $e: G_1 \times G_2 \rightarrow G_T$.

c) Choose a random generator P_1 of G_1.

d) Choose a random generator P_2 of G_2.

e) Choose four hash functions $H_1: \{0, 1\}^* \rightarrow G_1$, $H_2: \{0, 1\}^* \rightarrow Z_p$, $H_3: \{0, 1\}^* \rightarrow Z_p$, $H_4: \{0, 1\}^* \rightarrow Z_p$. An example of how to construct such hash functions is provided in Annex B.

f) Choose two random integers x, y in Z_p.

g) Compute $X = [x]P_2$ and $Y = [y]P_2$.

h) Output the following:

— group public parameter = $(G_1, G_2, G_T, e, P_1, P_2, p, H_1, H_2, H_3, H_4)$,

— group public key = (X, Y),

— group membership issuing key = (x, y).

NOTE Examples of recommended parameters are provided in Annex C.2.

The group membership issuing process requires a secure and authentic channel between the principal signer and the group membership issuer. How to establish such a channel is out scope of this mechanism. The group membership issuing process includes the following steps:

a) The group membership issuer chooses a nonce $n_I \in \{0, 1\}^t$.

b) The group membership issuer sends n_I to the principal signer.

c) The principal signer chooses a group member private key f randomly from Z_p or derives f from its secret seed value.

d) The principal signer computes $Q_2 = [f]P_1$.

e) The principal signer chooses u from Z_p and computes $U = [u]P_1$.

f) The principal signer computes $v = H_2(P_1 \,||\, Q_2 \,||\, U \,||\, X \,||\, Y \,||\, n_I)$.

g) The principal signer computes $w = (u + v \cdot f) \bmod p$.

h) The principal signer sends (Q_2, v, w) to the group membership issuer.

i) The group membership issuer computes $U' = [w]P_1 - [v]Q_2$.

j) The group membership issuer computes $v' = H_2(P_1 \,||\, Q_2 \,||\, U' \,||\, X \,||\, Y \,||\, n_I)$.

k) The group membership issuer verifies $v = v'$. If the verification fails, abort the group membership issuing process.

l) The group membership issuer chooses a random integer r from Z_p.

m) The group membership issuer computes $A = [r]P_1$, $B = [y]A$, $C = [x]A + [rxy]Q_2$.

n) The group membership issuer sets (A, B, C) as the signer's group membership credential and sends the credential to the principal signer.

o) The principal signer computes $D = [f]B$.

p) The principal signer sends (A, B, C, D) to the assistant signer.

q) The assistant signer verifies $e(A, Y) = e(B, P_2)$.

r) The assistant signer verifies $e(A + D, X) = e(C, P_2)$.

s) If the verification fails, the assistant signer aborts.

t) The group member signature key for the signer is (f, A, B, C).

6.5.3 Signature process

On input of a group public key (X, Y), a group member signature key (f, A, B, C), a linking base bsn, a nonce $n_V \in \{0, 1\}^t$, and a message $m \in \{0, 1\}^*$ to be signed, the signature process takes the following steps. The linking base is either a special symbol $\perp$ or an arbitrary string used for the linking capability.

NOTE 1 The nonce n_V is usually chosen by the verifier.

NOTE 2 Alternative way to handle n_V is to include n_V as part of the message m.

a) The principal signer has the group member private key f while the assistant signer has (A, B, C, D).

b) If $bsn = \perp$, the assistant signer chooses a random J from G_1, otherwise, computes $J = H_1(bsn)$.

c) The assistant signer chooses a random integer l from Z_p.

d) The assistant signer computes $R = [l]A$, $S = [l]B$, $T = [l]C$, and $W = [l]D$.

e) The assistant signer computes $c = H_3(R \,||\, S \,||\, T \,||\, W \,||\, n_V)$.

f) The assistant signer sends (c, J, S, m, bsn) to the principal signer.

g) The principal signer computes $K = [f]J$.

h) The principal signer chooses $n_T \in \{0, 1\}^t$.

i) The principal signer chooses a random integer r from Z_p.

j) The principal signer computes $R_1 = [r]J$ and $R_2 = [r]S$.

k) The principal signer computes $h = H_4(c \,||\, m \,||\, J \,||\, K \,||\, bsn \,||\, R_1 \,||\, R_2 \,||\, n_T)$.

l) The principal signer computes $s = (r + h \cdot f) \bmod p$.

m) The principal signer sends (K, h, s, n_T) to the assistant signer.

n) The assistant signer outputs the anonymous signature $\sigma = (R, S, T, W, J, K, h, s, n_V, n_T)$.

6.5.4 Verification process

On input of a message m, a linking base bsn, a nonce $n_V \in \{0, 1\}^t$, a signature $(R, S, T, W, J, K, h, s, n_V, n_T)$, a group public key (X, Y), the verification process takes the following steps:

a) If $bsn \neq \perp$, verify that $J = H_1(bsn)$.

b) Verify that $e(R, Y) = e(S, P_2)$ and $e(R + W, X) = e(T, P_2)$.

c) Compute $R_1 = [s]J - [h]K$.

d) Compute $R_2 = [s]S - [h]W$.

e) Verify that $h = H_4(H_3(R \,||\, S \,||\, T \,||\, W \,||\, n_V) \,||\, m \,||\, J \,||\, K \,||\, bsn \,||\, R_1 \,||\, R_2 \,||\, n_T)$.

f) Optionally, call the revocation checking process.

g) If any of the above verifications fails, output 0 (invalid), otherwise, output 1 (valid).

6.5.5 Linking process

Given two signatures $\sigma = (R, S, T, W, J, K, h, s, n_V, n_T)$ and $\sigma' = (R', S', T', W', J', K', h', s', n_V', n_T')$, the linking process takes the following step:

a) If $J = J'$ and $K = K'$, output 1 (linked), otherwise, output 0 (not linked).

NOTE If the linking process outputs 0 because of $J \neq J'$, it means that the linking process cannot determine whether two signatures were created by the same group member.

6.5.6 Revocation process

Details of the revocation process in this mechanism are surveyed in [10]. There are four types of revocation (private key revocation, verifier blacklist revocation, signature revocation, and credential update) supported in this mechanism. The first three revocations are the same in 6.4.6. The credential update revocation process is specified below. Private key revocation and signature revocation can be either global revocation or local revocation. Verifier blacklist revocation is a local revocation. Credential update is a global revocation.

Updating group public key process:

a) The group membership issuer chooses randomly x' in Z_p.

b) The group membership issuer computes $X' = [x']P_2$.

c) The new group public key is (X', Y).

d) The new group membership issuing key is (x', y).

Updating membership credential process:

a) The group membership issuer computes $\beta = x'/x \bmod p$.

b) For each legitimate member with group membership credential (A, B, C),

   1) The issuer computes $C' = [\beta]C$ and sends C' to the member.

   2) The member updates its credential as (A, B, C').

7 Mechanisms with opening capability

7.1 General

This clause specifies two digital signature mechanisms with opening capability. This type of digital signatures is called group signatures in the literature. In these mechanisms, there is an entity called group membership opener who is designated by the signer and who can identify the signer from a group signature. Each mechanism consists of key generation process, signature process, verification process, opening process, and revocation process.

NOTE The mechanisms and associated security proofs in 7.2 and 7.3 are based on [17] and [14], respectively.

7.2 Mechanism 5

7.2.1 Symbols

The following symbols apply in the specification of this mechanism.

— K_n, K, K_c, K_s: security parameters.

— p_1', p_2': prime numbers.

— p_1, p_2: $K_n/2$-bit prime numbers such that $p_1 = 2p_1' + 1$, $p_2 = 2p_2' + 1$.

— n: K_n-bit integer such that $n = p_1 p_2$.

— a_0, a_1, a_2, b, w: elements of QR(n).

— G: a group which DDH assumption holds.

— q: order of G.

— g: a generator of G.

— y_1, y_2: elements of Z_q.

— Y_1, Y_2: elements of G.

7.2.2 Key generation process

The key generation process has four parts: setup process, group-membership-issuer setup, group-membership-opener setup and group membership issuing process. The setup process outputs the group public parameter agreed by the entities involved. The group-membership-issuer setup outputs group public key and group membership issuing key (issuing key for short). The group-membership-opener setup outputs a public key of group membership opener, and group membership opening key. The group membership issuing process is an interactive protocol running between the group membership issuer and a user to create a group member signature key for the user.

The setup process outputs the following as group public parameter:

a) Choose the following parameters: K_n, K, K_c, K_s, K_e, $K_{e'}$.

b) Choose a hash function H: $\{0, 1\}^* \rightarrow \{0, 1\}^{Kc}$.

NOTE An example of recommended parameters is provided in Annex C.2.

The group-membership-issuer setup takes the following steps by the group membership issuer:

a) Choose an RSA modulus $n = p_1 p_2$ with $p_1 = 2p_1' + 1$, $p_2 = 2p_2' + 1$ such that p_1', p_2', p_1, p_2 are all primes and n has K_n bits.

b) Choose random elements $a_0, a_1, a_2, b, w \in \mathrm{QR}(n)$.

c) Output the following:

— group public key (gpk) = (n, a_0, a_1, a_2, b, w),

— group membership issuing key = (p_1, p_2).

The group-membership-opener setup takes the following steps by the group membership opener:

a) Choose a group G with order q which DHH assumption holds.

b) Choose a random generator g for group G.

c) Choose random elements: $y_1, y_2 \in Z_q$.

d) Compute $Y_1 = [y_1]g$, $Y_2 = [y_2]g$.

e) Output the following:

— group membership opener public key (opk) = (q, g, Y_1, Y_2),

— group membership opening key = (y_1, y_2).

The group membership issuing process between group membership issuer (issuer, for short) and user U_i is as follows:

a) User U_i chooses $x_i' \in \Lambda$ randomly, where Λ is the interval $(0, 2^{\lambda})$ where $\lambda = K_n + K + K_s$.

b) User U_i computes $C = a_1^{xi'} \bmod n$ and sends it together with a proof of its correct generation. An example of how to prove its correct generation is provided in Annex F.1.

c) Issuer verifies the proof.

d) Issuer chooses a random number $x_i'' \in \Lambda$ and sends it to user U_i.

e) User U_i verifies $x_i'' \in \Lambda$.

f) User U_i computes $x_i = (x_i' + x'') \bmod 2^{\lambda}$ and $(A_i', h_i) = (a_1^{xi} \bmod n, [x_i]g)$. User U_i sends them together with proofs of their correct generation. An example of how to prove its correct generation is provided in Annex F.2.

g) Issuer verifies the proofs.

h) Issuer chooses a prime number $e_i' \in \{0, 1\}^{Ke'}$ where $e_i = 2^{Ke-1} + e_i'$ is also prime.

i) Issuer computes $A_i = (a_0 A_i')^{1/ei} \bmod n$ and $B_i = b^{1/ei'} \bmod n$.

j) Issuer stores $(i, (A_i, e_i', B_i, h_i))$ in member-list LIST.

k) Issuer sends (A_i, e_i', B_i) to user U_i.

l) User U_i computes $e_i = 2^{Ke-1} + e_i'$ and verifies both e_i' and e_i are primes.

m) User U_i verifies $a_0 a_1{}^{xi} \equiv A_i{}^{ei}$ and $b \equiv B_i{}^{ei'} \bmod n$.

n) User U_i stores (A_i, e_i', B_i, h_i) as his group membership credential and x_i as his group membership private key. The group member signature key for U_i is $(x_i, A_i, e_i', B_i, h_i)$.

7.2.3 Signature process

On input of a group public key gpk = (n, a_0, a_1, a_2, b, w), a group membership credential (A_i, e_i', B_i, h_i), a group membership private key x_i, a group membership opener public key opk = (q, g, Y_1, Y_2) agreed with the verifier, and a message $M \in \{0, 1\}^*$ to be signed, the signature process takes the following steps by signer U_i:

a) Choose $\rho_E \in Z_q$ and compute $\mathbf{E} = (E_0, E_1, E_2) = ([\rho_E]g, h_i + [\rho_E]Y_1, h_i + [\rho_E]Y_2)$.

b) Choose $\rho_m \in \{0, 1\}^{Kn/2}$ randomly and compute $A_{COM} = A_i a_2{}^{\rho m} \bmod n$ and $s = e_i \rho_m$ where $e_i = 2^{Ke-1} + e_i'$.

c) Choose $\rho_r \in \{0, 1\}^{Kn/2}$ randomly and compute $B_{COM} = B_i w^{\rho r} \bmod n$ and $t = e_i' \rho_r$.

d) Choose $\mu_x \in \{0, 1\}^{\lambda + Kc + Ks}$, $\mu_s \in \{0, 1\}^{Ke + (Kn/2) + Kc + Ks}$, $\mu_{e'} \in \{0, 1\}^{Ke' + Kc + Ks}$, $\mu_t \in \{0, 1\}^{Ke' + (Kn/2) + Kc + Ks}$, and $\mu_E \in Z_q$ randomly.

e) Compute $V_{ComCipher} = (V_{ComCipher0}, V_{ComCipher1}, V_{ComCipher2}) = ([\mu_E]g, [\mu_x]g + [\mu_E]Y_1, [\mu_x]g + [\mu_E]Y_2)$.

f) Compute $V_{ComMPK} = a_1{}^{\mu x} a_2{}^{\mu s} A_{COM}{}^{-\mu e'} \bmod n$.

g) Compute $V_{ComRev} = w^{\mu t} B_{COM}{}^{-\mu e'} \bmod n$.

h) Compute $c = H(K_n \| K_e \| K_{e'} \| K \| K_c \| K_s \| \text{gpk} \| \text{opk} \| \mathbf{E} \| A_{COM} \| B_{COM} \| V_{ComCipher} \| V_{ComMPK} \| V_{ComRev} \| M)$.

i) Compute $\tau_x = cx_i + \mu_x$, $\tau_s = cs + \mu_s$, $\tau_t = ct + \mu_t$, $\tau_{e'} = ce_i' + \mu_{e'}$, $\tau_E = c\rho_E + \mu_E \bmod q$.

j) Verify $|\tau_x| \le \lambda + K_c + K_s$ and $|\tau_{e'}| \le K_{e'} + K_c + K_s$. If they do not hold, go to step d).

k) Output $\sigma = (\mathbf{E}, A_{COM}, B_{COM}, c, \tau_x, \tau_s, \tau_{e'}, \tau_t, \tau_E)$ as signature for the message M.

7.2.4 Verification process

On input of a message M, a signature $\sigma = (\mathbf{E}, A_{COM}, B_{COM}, c, \tau_x, \tau_s, \tau_{e'}, \tau_t, \tau_E)$, a group public key gpk = (n, a_0, a_1, a_2, b, w), a group membership opener public key opk = (q, g, Y_1, Y_2), the verification process takes the following steps:

a) Verify $|\tau_x| \le \lambda + K_c + K_s$ and $|\tau_{e'}| \le K_{e'} + K_c + K_s$ hold.

b) Compute $\tau_e = \tau_{e'} + c \cdot 2^{Ke-1}$, $V'_{ComCipher} = (V'_{ComCipher0}, V'_{ComCipher1}, V'_{ComCipher2}) = ([\tau_E]g - [c]E_0, [\tau_x]g + [\tau_E]Y_1 - [c]E_1, [\tau_x]g + [\tau_E]Y_2 - [c]E_2)$, $V'_{ComMPK} = a_0{}^c a_1{}^{\tau x} a_2{}^{\tau s} A_{COM}{}^{-\tau e} \bmod n$, and $V'_{ComRev} = b^c w^{\tau t} B_{COM}{}^{-\tau e'} \bmod n$.

c) Verify $c = H(K_n \| K_e \| K_{e'} \| K \| K_c \| K_s \| \text{gpk} \| \text{opk} \| \mathbf{E} \| A_{COM} \| B_{COM} \| V'_{ComCipher} \| V'_{ComMPK} \| V'_{ComRev} \| M)$.

d) If any of the above verifications fails, output 0 (invalid), otherwise, output 1 (valid).

7.2.5 Opening process

Given a signature $\sigma = (\mathbf{E}, A_{COM}, B_{COM}, c, \tau_x, \tau_s, \tau_{e'}, \tau_t, \tau_E)$, the group membership opening process takes the following steps by the group membership opener with group membership opening key (y_1, y_2):

a) Compute $S_1 = E_1 - [y_1]E_0$ and $S_2 = E_2 - [y_2]E_0$. Verify $S_1 = S_2$ and put $h = S_1$.

b) Search for h in the member-list LIST and output corresponding User ID.

c) Else output Opening failure.

7.2.6 Revocation process

When a user is leaving the group or have his/her membership revoked, group membership issuer needs to update group public key, and other users needs to update his membership credential in order to generate signatures consistent with the new group public key. This revocation process is a global revocation.

Updating group public key process:

Let mpk' = ($\underline{A}$, $\underline{e}$, $\underline{B}$, $\underline{h}$) be the group membership credential of a leaving user, the group membership revocation process takes the following steps by the group membership issuer with group membership issuing key (p_1, p_2):

a) Compute $b' = b^{1/\underline{e}} \bmod n$.

b) Update group public key gpk to $(n, a_0, a_1, a_2, b', w)$ and the revocation list RL to RL ∪ {(mpk', gpk)}.

Updating membership credential process:

Given $mpk_i = (A_i, e_i', B_i, h_i)$ and (mpk' = ($\underline{A}$, $\underline{e}$, $\underline{B}$, $\underline{h}$), gpk = $(n, a_0, a_1, a_2, b', w)$) from the revocation list RL, the updating membership credential process of mpk_i takes the following steps:

a) Compute α, β such that $\alpha\underline{e} + \beta e_i' = 1$.

b) Compute $B_i' = B_i^{\alpha} b^{\beta} \bmod n$ and replace mpk_i to (A_i, e_i', B_i', h_i).

7.3 Mechanism 6

7.3.1 Symbols

The following symbols apply in the specification of this mechanism.

— $P_1, Q_1, R_1, U, U', A, B$: elements of G_1.

— P_2, Y: elements of G_2.

— $P_3, S, T, W, W', Z, V, Z', V', W'$: Elements in G_3.

— Y': elements of G_T.

— $x, s, t, f, c', c, d, f', u', r, q, y, v, z, c, y', z', h', g', u, h, g, com, a, b, j, n, o$: integers in Z_p.

— H: a hash function that output elements in Z_p.

7.3.2 Key generation process

The key generation process has two parts: setup process and group membership issuing process. The setup process is executed by the group membership issuer to create the group public parameter, group public key, and group membership issuing key. The group membership issuing process is an interactive protocol running between the group membership issuer and a group member to create a unique group member signature key for the group member.

The setup process takes the following steps by the group membership issuer:

a) Choose an asymmetric bilinear group pair (G_1, G_2) of large prime order p and an associated pairing function $e: G_1 \times G_2 \rightarrow G_T$.

b) Choose a cyclic group G_3 of the prime order p such that the decision Diffie-Hellman problem is difficult. If G_3 is an elliptic curve, it is supposed to be defined on the field whose characteristic is different from those on which G_1 and G_2 are defined.

c) Choose a random generator P_1, Q_1, R_1 of G_1.

d) Choose a random generator P_2 of G_2.

e) Choose a random generator P_3 of G_3.

f) Choose a hash function $H: \{0, 1\}^* \rightarrow Z_p$. An example of how to construct such hash function is provided in Annex B.

g) Choose three random integers x, s, t in Z_p.

h) Compute $X = [x]P_2$, $S = [s]P_3$, and $T=[t]P_3$.

i) Output the following:

— group public parameter = $(G_1, G_2, G_T, e, G_3, p, H)$,

— group public key = $(P_1, Q_1, R_1, P_2, P_3, X, S, T)$,

— opening key = (s, t),

— group membership issuing key = (x).

NOTE Examples of recommended parameters are provided in Annex C.2.

The group membership issuing process requires a secure and authentic channel between the signer and the group membership issuer. How to establish such a channel is out scope of this mechanism. The group membership issuing process includes the following steps:

a) The signer chooses f randomly from Z_p or derives f from its secret seed value, where f is part of the group member private key.

b) The signer computes $W = [f]P_3$.

c) The signer chooses u from Z_p and computes $U = [f]Q_1+[u]R_1$.

d) The signer sends (W, U) to the group membership issuer.

e) The group membership issuer randomly chooses c' and d from Z_p and computes $com = H(c' \,||\, d)$.

f) The group membership issuer sends com to the signer.

g) The signer randomly chooses f' and u' and computes $W' = [f']P_3$ and $U' = [f']Q_1+[u']R_1$.

h) The signer sends (W', U') to the group membership issuer.

i) The group membership issuer sends (c', d) to the signer.

j) The signer verifies $com = H(c' \,||\, d)$. If the verification fails, aborts the group membership issuing process.

k) The signer computes $r=f\,c'+f'$ mod p and $q=u\,c'+u'$ mod p.

l) The signer sends (r, q) to the group membership issuer.

m) The group membership issuer verifies $[r]P_3=[c']W+W'$ and $[r]Q_1+[q]R_1=[c']U+U'$. If the verification fails, abort the group membership issuing process.

n) The group membership issuer chooses a random integer y and v from Z_p.

o) The group membership issuer computes $A = [1/(x+y)](P_1 - U - [v]R_1)$.

p) The group membership issuer sets (A, y) as the signer's group membership credential and sends (A, y, v) to the signer.

q) The signer computes $z = u+v$, and sets (A, y) as its group membership credential.

r) The signer verifies $e(A, X+[y]P_2) \cdot e([f]Q_1,P_2) \cdot e([z]R_1,P_2) = e(P_1, P_2)$.

s) If the verification fails, the signer aborts.

t) The group member signature key for the signer is (f, A, y, z).

7.3.3 Signature process

On input of a group public key $(P_1, Q_1, R_1, P_2, P_3, X, S, T)$, a group member signature key (f, A, y, z), and a message $m \in \{0, 1\}^*$ to be signed, the signature process takes the following steps.

a) The signer has the group member signature key (f, A, y, z).

b) The signer chooses a random integer g and h from Z_p.

c) The signer computes $B = A+[h]R_1$, $Z = [f+g]P_3$, $V = [g]S$, and $W = [g]T$.

d) The signer randomly chooses $a, b, j, n,$ and o from Z_p.

e) The signer computes $Y = e(Q_1, P_2)^a \cdot e(B, P_2)^b \cdot e(R_1, P_2)^j \cdot e(R_1, X)^n$, $Z' = [a+o]P_3$, $V' = [o]P_3$, and $W' = [o]P_3$.

f) The signer computes $c = H(p \,||\, P_1 \,||\, P_2 \,||\, P_3 \,||\, X \,||\, S \,||\, T \,||\, Q_1 \,||\, R_1 \,||\, B \,||\, Z \,||\, V \,||\, W \,||\, Y \,||\, V' \,||\, W' \,||\, Z' \,||\, m)$.

g) The signer computes $f' = cf + a \bmod p$, $y' = cy + b \bmod p$, $z' = c(z - h\,y)+j \bmod p$, $h' = -c\,h + n \bmod p$, and $g' = c\,g+o \bmod p$.

h) The signer outputs the group signature $\sigma = (B, Z, V, W, c, f', y', z', h', g')$.

7.3.4 Verification process

On input of a message m, a signature $\sigma = (B, Z, V, W, c, f', y', z', h', g')$, a group public key $(P_1, Q_1, R_1, P_2, P_3, X, S, T)$, the verification process takes the following steps:

a) Compute $Y = e(Q_1, P_2)^{f'} \cdot e(B, [y']P_2 + [c]X) \cdot e(R_1, P_2)^{z'} \cdot e(R_1, X)^{h'} \cdot e(P_1, P_2)^{-c}$, $Z' = [f'+g']P_3 - [c]Z$, $V' = [g']S - [c]V$, and $W' = [g']T - [c]W$.

b) Verify that $c = H(p \,||\, P_1 \,||\, P_2 \,||\, P_3 \,||\, X \,||\, S \,||\, T \,||\, Q_1 \,||\, R_1 \,||\, B \,||\, Z \,||\, V \,||\, W \,||\, Y \,||\, V' \,||\, W' \,||\, Z' \,||\, m)$ holds.

c) If the above verifications fails, output 0 (invalid), otherwise, output 1 (valid).

7.3.5 Opening process

Given a signature $\sigma = (B, Z, V, W, c, f', y', z', h', g')$, the opening process takes the following steps:

a) If the verification fails, outputs $\perp$ (failure).

b) Compute $W = Z - [1/s]V$.

c) Output W.

7.3.6 Revocation process

This revocation process is a global revocation.

a) The group membership issuer is given a revoked signer's group membership credential $(\underline{A}, \underline{y})$.

b) The group membership issuer computes $\underline{Q}_1 = [1/(x+\underline{y})]Q_1$, $\underline{P}_1 = [1/(x+\underline{y})]P_1$, and $\underline{R}_1 = [1/(x+\underline{y})]R_1$.

c) The group membership issuer sends $(\underline{y}, \underline{Q}_1, \underline{P}_1, \underline{R}_1)$ to each signer.

d) Each signer whose group member signature key is (f, A, y, z) compute $A' = [1/(\underline{y}-y)](A-\underline{P}_1-[f]\underline{Q}_1-[z]\underline{R}_1)$ and sets its group member signature key as (f, A', y, z).

e) Each signer sets the group public key as $(\underline{P}_1, \underline{Q}_1, \underline{R}_1, P_2, P_3, X, S, T)$.

8 Mechanisms with both opening and linking capabilities

8.1 General

This clause specifies a digital signature mechanism with both opening capability and linking capability. This type of digital signatures is called group signatures with controllable linkability in the literature. In the mechanism, there is an entity called group membership opener who is designated by the signer and who can identify the signer from a group signature. There is also a group signature linker who can determine whether two group signatures were created by the same group member.

NOTE The mechanism and associated security proofs in 8.2 are based on [15] and [16].

8.2 Mechanism 7

8.2.1 Symbols

The following symbols apply in the specification of this mechanism.

— $Q, Q_1, Q_2, U, W, D, V, A, Z, W_{ID}, Q_1', Q_2', U', W', D', \tilde{A}, D_1, D_2, D_3, R_1, R_2, R_3, K_{\text{open}}, W_{\text{open}}, V_{\text{open}}, Y_{1,i}, Y_{2,i}, X_{1,i}, X_{2,i}, S_{1,j}, S_{2,j}, S_{3,j}, S_{4,j}, S_{5,j}$: elements of G_1.

— B_1, B_θ: elements of G_2.

— $L_1, L_2, L_3, L_4, L_A, L_1', L_2', L_3', L_4'$: elements of G_T.

— $\eta, \xi, \theta, x, y, z, r_{ID}, c_{ID}, s_{ID}, \alpha, \gamma, r_\alpha, r_x, r_\gamma, r_y, c, s_\alpha, s_x, s_\gamma, s_y, c_{\text{open}}, s_{\text{open}}, x_1, x_2, x_j, v_j$: integers in Z_p^*.

— $i, j, \lambda, \rho, \kappa, t$: t-bit integers where t is a fixed non-negative integer.

— H_p: a hash function that outputs elements in Z_p^*.

The mechanism consists of key generation process (including a setup process and a group membership issuing process), signature process, verification process, opening process, linking process, and revocation process. An evidence evaluation process is used to verify the validity of an evidence of binding generated from opening process. A group signature revocation process has three sub-processes, Gen-RL, Update-gpk, and Update-usk.

8.2.2 Key generation process

The key generation process has two parts: a setup process and a group membership issuing process.

The setup process takes as input a security parameter and generates group public key *gpk* and its corresponding group membership issuing key *gmik*, group membership opening key *gmok*, and group signature linking key *gslk* as follows:

Generate group public parameter as follows:

a) Generate three groups G_1, G_2, G_T of prime order p and a bilinear map $e: G_1 \times G_2 \rightarrow G_T$. Assume that the groups are multiplicative.

b) Choose $B_1 \leftarrow G_2$ and $Q_1, Q_2, Q, U \leftarrow G_1$.

c) Choose a cryptographic hash function $H_p: \{0, 1\}^* \rightarrow Z_p^*$, where $H_p(M)$ is the hash-code of message $M \in \{0, 1\}^*$. An example of how to construct such hash function is provided in Annex B.

Generate keys as follows.

a) Choose $\eta, \xi, \theta \leftarrow Z_p^*$.

b) Compute $W = [\eta]U$, $D = [\xi]U$, $B_\theta = [\theta]\, B_1$, $V = [\xi]\, B_1$.

c) Compute $L_1 = e(W, B_1)$, $L_2 = e(W, B_\theta)$, $L_3 = e(Q_1, B_1)$, and $L_4 = e(Q_2, B_1)$.

d) Output group public parameter = $((e, G_1, G_2, G_T), Q, B_1, B_\theta, H_p)$.

e) Output the initial group public key $gpk = (Q_1, Q_2, U, W, D, (L_1, L_2, L_3, L_4))$.

f) Output $gmik = \theta$, $gmok = (\eta, \xi)$, and $gslk = (V)$.

NOTE 1 L_1, L_2, L_3, and L_4 are optional in gpk, because they can be computed from Q_1, Q_2, B_1, B_θ, W by the signers and verifiers.

NOTE 2 Examples of recommended parameters are provided in Annex C.2.

The secret authority keys *gmik*, *gmok*, and *gslk* are kept secret by the group membership issuer, the group membership opener, and group signature linker, respectively.

The group membership issuing process is as follows.

The group membership issuer manages a member list **LIST**=(LIST[1],...,LIST[n]) where n is the number of group members who are registered so far. Each entry of the list contains private information associated with each registered user.

Two sub-processes, UserJoin (run by a joining user with identity *ID*) and Issue (run by the group membership issuer) interactively generate a group member signature key as follows. Assume that the two sub-processes communicates via a secure authentication channel.

— UserJoin proceeds as follows:

  a) Choose a random group member private key $sk = z \leftarrow Z_p^*$ and compute $Z = [z]W$.

  b) Choose $r_{ID} \leftarrow Z_p^*$ and compute $W_{ID} = [r_{ID}]W$.

  c) Compute $c_{ID} = H_p(ID \,||\, W \,||\, Z \,||\, W_{ID})$.

  d) Compute $s_{ID} = r_{ID} + c_{ID}\, z \pmod p$.

  e) $T_{ID} = (Z, s_{ID}, c_{ID})$.

  f) Send ($Join\_Request$, ID, T_{ID}) to Issue process.

  NOTE To be continued to g) after the Issue process.

— Issue proceeds as follows:

  a) Receive a join-request message ($Join\_Request$, ID, T_{ID}).

  b) Check the validity of the message as follows.

    1) Check if ID is valid.

2) Check if $c_{ID} = H_p(ID \,||\, W \,||\, Z \,||\, [s_{ID}]W\text{-}[c_{ID}]Z)$.

c) Find i with LIST[i]=(ID, ·, ·, ·, ·, ·, ·) containing the ID from the member-list **LIST**= (LIST[1],..., LIST[n]).

d) If ID has not been registered, i.e., no matching i exists then proceed as follows.

1) Choose $x, y \leftarrow Z_p^*$.

2) Compute $A = [1/(\theta+x)](Q_1\text{-}[y]Q_2\text{-}Z)$ (=$[1/(\theta+x)](Q_1\text{-}[y]Q_2\text{-}[z]W)$).

3) Add LIST[n+1] = (ID, $[y]Q$, A, x, y, $upk[i] = Z$(=$[z]W$), T_{ID}) to **LIST**.

Otherwise,

1) Choose $x \leftarrow Z_p^*$.

2) Compute $A = [1/(\theta+x)](Q_1\text{-}[y]Q_2\text{-}Z)$ (=$[1/(\theta+x)](Q_1\text{-}[y]Q_2\text{-}[z]W)$).

3) Replace the previous LIST[i] with new LIST[i]= (ID, $[y]Q$, A, x, y, $upk[i] = Z$, T_U).

e) Send the group membership credential (i, A, x, y) to UserJoin algorithm.

— UserJoin continues as follows:

NOTE Continued from (f) before the Issue process.

g) Receive message (i, A, x, y).

h) Check if $e(A, B_\theta+[x]B_1) = e(Q_1\text{-}[y]Q_2\text{-}[z]W, B_1)$ where Q_1, Q_2, W are included in gpk. If the equality does not hold then abort. Otherwise, the i^{th} group member signature key is usk_{i0} = (0, x, y, z, $A = [1/(\theta+x)](Q_1\text{-}[y]Q_2\text{-}[z]W)$). The value 0 means that this key is a group member signature key which is initially issued by the group membership issuer. This key will be updated by revocation and 0 will be changed into an index according to the update.

8.2.3 Signature process

A group member signature key includes an index κ to indicate that the key has been updated up to the κ^{th} entry of the revocation index list **RI** (refer to the group signature revocation process). Let λ ($\geq \kappa$) be the most up-to-date number of revoked keys in **RI**. To generate a signature, the group member signature key is updated up to the λ^{th} entry of **RI**. A generated signature includes λ to indicate that the signature was generated with the key that has been updated up to the λ^{th} entry of **RI**. The signature can be verified with a group public key that has been updated up to the λ^{th} entry of **RI**.

The signature process takes as input group public parameter ((e, G_1, G_2, G_T), Q, B_1, B_θ, H_p), a group public key gpk_κ, a group member signature key $usk_{i\kappa}$ = (κ, x, y, z, A), and a message $M \in \{0,1\}^*$ where κ ($\leq \lambda$) is the last updated revocation index for the signer i and λ is the up-to-date revocation index among all group members. It then proceeds as follows.

a) Call Update-usk of the group signature revocation process with $usk_{i\kappa}$, and obtain gpk_λ =(Q_1', Q_2', U', W', D', (L_1', L_2', L_3', L_4')), and an updated group member signature key $usk_{i\lambda}$ =(λ, x, y, z, $\tilde{A}$) where $\tilde{A}$ has been updated up to the λ^{th} entry of **RI**.

b) Choose $\alpha \leftarrow Z_p^*$.

c) Compute $D_1 = [\alpha]U'$, $D_2 = \tilde{A}+[\alpha]W'$, $D_3 = [y]Q+[\alpha]D'$.

d) Compute $\gamma = x\alpha - z \pmod p$.

e) Choose r_α, r_x, r_γ, $r_y \leftarrow Z_p^*$.

f) Compute $R_1 = [r_\alpha]U'$.

g) Compute $R_2 = e(D_2, B_1)^{rx}\, e(W', B_\theta)^{-r\alpha}\, e(W', B_1)^{-r\gamma}\, e(Q_2', B_1)^{ry}$.

h) Compute $R_3 = [r_y]Q + [r_\alpha]D'$.

i) Compute $c = H_p(M \,||\, \lambda \,||\, D_1 \,||\, D_2 \,||\, D_3 \,||\, R_1 \,||\, R_2 \,||\, R_3)$.

j) Compute $s_\alpha = r_\alpha + c\alpha \pmod p$, $s_x = r_x + cx \pmod p$, $s_\gamma = r_\gamma + c\gamma \pmod p$, $s_y = r_y + cy \pmod p$.

k) Output $\sigma = (\lambda, D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$.

NOTE Using the pre-computed $L_A = e(\tilde{A}, B_1)$, Step g) can be modified as follows: Compute $R_2 = L_A{}^{rx} \cdot (L_1')^{\alpha rx - r\gamma} \cdot (L_2')^{-r\alpha} \cdot (L_4')^{-ry}$.

8.2.4 Verification process

To verify a signature with index ρ, the group public key updated up to the ρ^{th} entry of **RL** is used. Note that ρ might not be the most up-to-date number of revoked keys in **RL**, because some keys could be revoked before verifying.

The group signature verification process takes as input group public parameter $((e, G_1, G_2, G_T), Q, B_1, B_\theta, H_p)$, $gpk_0 = (Q_1, Q_2, U, W, D, (L_1, L_2, L_3, L_4))$, a signature $\sigma = (\rho, D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$ and a message $M \in \{0,1\}^*$ and then proceeds as follows. Here ρ is the revocation index at the time of signature generation.

a) Call Update-gpk of Revocation with gpk, $(\rho, \mathbf{RL})$, and obtain $gpk_\rho = (Q_1', Q_2', U', W', D', (L_1', L_2', L_3', L_4'))$.

b) Compute $R_1 = [s_\alpha]U' - [c]D_1$.

c) Compute $R_2 = e(D_2, B_1)^{sx}\, e(W', B_\theta)^{-s\alpha}\, e(W', B_1)^{-s\gamma}\, e(Q_2', B_1)^{sy}\, (e(D_2, B_\theta)/e(Q_1', B_1))^c$.

d) Compute $R_3 = [s_y]Q + [s_\alpha]D' - [c]\, D_3$.

e) Checks if the equality $c = H_p(M \,||\, \rho \,||\, D_1 \,||\, D_2 \,||\, D_3 \,||\, R_1 \,||\, R_2 \,||\, R_3)$ holds.

f) If the equality holds then output 1 (valid). Otherwise, output 0 (invalid).

NOTE To minimize pairing computation, Step c) can be modified as follows: Compute $R_2 = e(D_2, [s_x]\, B_1 + [c]\, B_\theta)\, L_2'^{-s\alpha} L_1'^{-s\gamma} L_4'^{\,sy} L_3'^{-c}$.

8.2.5 Opening process

Group membership opening process takes as input a signature $\sigma = (\rho, D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$, and the group membership opening key $gmok = (\eta, \xi)$ and proceeds as follows:

a) Compute $[y]Q = D_3 - [\xi]D_1$.

b) Find i with $\text{LIST}[i] = (ID, [y]Q, A, x, y, upk[i]=Z(=[z]W), \cdot)$ from the member-list **LIST**.

c) If no matching i exists then output $(i = 0, *)$. Otherwise, it proceeds as follows:

— Choose $r \leftarrow Z_p^*$.

— Compute $K_{\text{open}}=[\eta]D_1$, $W_{\text{open}}=[r]U$, $V_{\text{open}}=[r]D_1$, $c_{\text{open}}= H_p(\sigma \,||\, g \,||\, K_{\text{open}} \,||\, W_{\text{open}} \,||\, V_{\text{open}})$, and $s_{\text{open}}= r + c_{\text{open}}\, \eta \pmod p$.

— Output an evidence of binding $(i, \tau = (K_{\text{open}}, c_{\text{open}}, s_{\text{open}}), upk[i]=Z, Y_{1,i}= [y_i]Q_2, Y_{2,i}=[y_i]B_1, X_{1,i}=[x_i]Q, X_{2,i}=[x_i]B_1)$.

— If evidence evaluation process with this evidence of binding outputs 1, output User ID corresponding to i.

8.2.6 Evidence evaluation process

Evidence evaluation process takes as input a group public key gpk, a signature $\sigma = (\rho, D_1, D_2, D_3, c, s_\alpha, s_x, s_\gamma, s_y)$, an evidence of binding $(i, \tau = (K_{open}, c_{\text{open}}, s_{\text{open}}), upk[\text{i}] = Z, Y_{1,i}, Y_{2,i}, X_{1,i}, X_{2,i})$ and then proceeds as follows:

a) Call Update-gpk of Revocation with gpk, $(\rho, \mathbf{RL})$, and obtain $gpk_\rho = (Q_1', Q_2', U', W', D', (L_1', L_2', L_3', L_4'))$.

b) If $i = 0$ then output $\perp$ (failure).

Otherwise, check if the following equalities hold:

— $c_{\text{open}} = H_p(\sigma \,||\, g \,||\, K_{\text{open}} \,||\, [s_{\text{open}}]U - [c_{\text{open}}]W \,||\, [s_{\text{open}}]D_1 - [c_{\text{open}}]K_{\text{open}})$.

— $e(D_2 - K_{open}, X_{2,i} + B_\theta) = e(Q_1 - Y_{1,i} - Z_i, B_1')$ where Q_1 is included in gpk and $\log_{B1}B_1' = \log_{Q1}Q_1'$,

If all the equalities hold then output 1 ("valid"), and otherwise, 0 ("invalid").

NOTE Assume that $e(X_{1,i}, B_1) = e(Q, X_{2,i})$ and $e(Y_{1,i}, B_1) = e(Q_2, Y_{2,i})$ where Q and Q_2 are included in gpk.

8.2.7 Linking process

Group signature linking process takes as input two signatures σ and σ', and the group signature linking key $gslk = (B_1, V = [\xi]\, B_1)$ and then proceeds as follows.

a) Compute $LI_1 = e(D_3, B_1)\, e(D_1, V)^{-1}$ and $LI_2 = e(D_3', B_1)\, e(D_1', V)^{-1}$.

b) If $LI_1 = LI_2$ then output 1 (linked), and otherwise, 0 (not linked).

NOTE Alternatively, Step a) can be modified as follows: Compute $LI_1 = e(D_3 - D_3', B_1)$ and $LI_2 = e(D_1 - D_1', V)$.

8.2.8 Revocation process

RI is the revocation index list that contains all indices of the revoked group members so far. Whenever keys to be revoked are given, **RI** is immediately updated to include them. **RL** is the list that contains private information of the revoked group members so far. Assume that **RL** is always updated to the latest revocation index of **RI**. Anybody can publicly make use of **RL** and **RI**. The lists **RI** and **RL** are initially set to be empty.

This revocation process is a global revocation. It performs three sub-processes, Gen-RL (run by Issuer), Update-gpk (run by any party) and Update-usk (run by a valid signer or group member).

— Gen-RL:

  a) Assume that **RI** = $\{j_1, \ldots, j_\lambda\}$ is given. Let LIST[i]= (ID, $[y_i]Q$, A_i, x_i, y_i, $\cdot$, $\cdot$) in the member-list **LIST**. Define $\upsilon_k = (\theta - x_{j1})(\theta - x_{j2})\ldots(\theta - x_{jk}) \pmod p$.

  b) For each $j \in$ **RI**,

    1) Compute $S_{1,j} = [1/\upsilon_j]Q_1$, $S_{2,j} = [1/\upsilon_j]Q_2$, $S_{3,j} = [1/\upsilon_j]U$, $S_{4,j} = [1/\upsilon_j]W$, $S_{5,j} = [1/\upsilon_j]D$.

    2) Add $(S_{1,j}, S_{2,j}, S_{3,j}, S_{4,j}, S_{5,j}, x_j)$ to **RL**

  c) Publish **RL**=$\{(S_{1,j}, S_{2,j}, S_{3,j}, S_{4,j}, S_{5,j}, x_j) \mid j \in$ **RI**$=\{j_1, \ldots, j_\lambda\}\}$.

— Update-gpk:

  a) It takes as input initial gpk, index ρ, and **RL**.

  b) Obtain λ from **RL**, where λ is the number of all revoked keys so far.

  c) If $\lambda < \rho$ or $\rho < 0$ then abort.

d) Otherwise, i.e., $0 \le \rho < \lambda$, then update the group public key up to the ρ^{th} revoked key.

1) Set $\mathbf{RI}(\rho) = \{j_1,..., j_\rho\} \subseteq \mathbf{RI}$ and $\mathbf{RL}(\rho)=\{(S_{1,j}, S_{2,j}, S_{3,j}, S_{4,j}, S_{5,j}, x_j) | j \in \mathbf{RI}=\{j_1,..., j_\rho\}\}$

2) Compute $Q_1'= S_{1,j\rho}$, $Q_2'= S_{2,j\rho}$, $U'= S_{3,j\rho}$, $W'= S_{4,j\rho}$ and $D'= S_{5,j\rho}$.

3) Compute $L_1'= e(W', B_1)$, $L_2 = e(W', B_\theta)$, $L_3 = e(Q_1', B_1)$, and $L_4' = e(Q_2', B_1)$.

4) Output an updated group public key $gpk_\rho = (Q_1', Q_2', U', W', D', (L_1', L_2', L_3', L_4'))$.

— Update-usk:

a) It takes as input $usk_{i\kappa} = (\kappa, x, y, z, A)$.

b) Compute gpk_λ by calling Update-gpk with $(gpk,(-1, \mathbf{RL}))$.

c) Set $\mathbf{RI}(\kappa,\lambda)= \{j_{\kappa+1},..., j_\lambda\} \subseteq \mathbf{RI}$.

d) Compute $\tilde{A} = [(-1)^{\lambda-\kappa}/\pi_{\lambda-\kappa}]A + (\prod_{j\in\mathbf{RI}(\kappa,\lambda)} [(-1)^{\lambda-\kappa-j}\pi_{j-1}/\pi_{\lambda-\kappa}](S_{1,j}+[-y]S_{2,j}+[-z]S_{3,j}))$ where $S_{1,j}$, $S_{2,j}$, $S_{3,j}$, $x_j \in \mathbf{RL}$, and $\pi_0=1$ and $\pi_j = (x-x_1)(x-x_2)...(x-x_j)$ (mod p).

e) Output $(gpk_\lambda, usk_{i\lambda} =(\lambda, x, y, z, \tilde{A}))$.

Annex A
(normative)

Object identifiers

This annex specifies the object identifiers using ASN.1 module for all the mechanisms specified in this part of ISO/IEC 20008.

```
AnonymousSignatureUsingGroupPK {
   iso(1) standard(0) anonymous-digital-signatures(20008) part2(2)
      asn1-module(1) mechanisms-using-group-public-key(0) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- EXPORTS All; --

IMPORTS

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

OID ::= OBJECT IDENTIFIER -- alias

-- Synonyms --

id-as-gpk OID ::= {
   iso(1) standard(0) anonymous-digital-signatures(20008) part2(2)
      algorithm(0) }

-- Assignments --

id-as-gpk-m-1 OID ::= { id-as-gpk mechanism1(1) }
id-as-gpk-m-2 OID ::= { id-as-gpk mechanism2(2) }
id-as-gpk-m-3 OID ::= { id-as-gpk mechanism3(3) }
id-as-gpk-m-4 OID ::= { id-as-gpk mechanism4(4) }
id-as-gpk-m-5 OID ::= { id-as-gpk mechanism5(5) }
id-as-gpk-m-6 OID ::= { id-as-gpk mechanism6(6) }
id-as-gpk-m-7 OID ::= { id-as-gpk mechanism7(7) }

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

ASAlgorithms ALGORITHM ::= {
   as-gpk-m-1 |
   as-gpk-m-2 |
   as-gpk-m-3 |
   as-gpk-m-4 |
   as-gpk-m-5 |
   as-gpk-m-6 |
   as-gpk-m-7
}

as-gpk-m-1 ALGORITHM ::= {
   OID id-as-gpk-m-1 PARMS HashFunctions
}

as-gpk-m-2 ALGORITHM ::= {
   OID id-as-gpk-m-2 PARMS HashFunctions
}

as-gpk-m-3 ALGORITHM ::= {
   OID id-as-gpk-m-3 PARMS HashFunctions
```

```
}

as-gpk-m-4 ALGORITHM ::= {
   OID id-as-gpk-m-4 PARMS HashFunctions
}

as-gpk-m-5 ALGORITHM ::= {
   OID id-as-gpk-m-5 PARMS HashFunctions
}

as-gpk-m-6 ALGORITHM ::= {
   OID id-as-gpk-m-6 PARMS HashFunctions
}

as-gpk-m-7 ALGORITHM ::= {
   OID id-as-gpk-m-7 PARMS HashFunctions
}

-- Cryptographic algorithm identification --

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

END -- AnonymousSignatureUsingGroupPK --
```

Annex B
(normative)

Special hash-functions

B.1 Converting between bit strings and integers: BS2IP and I2BSP

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 $x = \langle x_{l-1}, \ldots, x_0 \rangle$ where x_0, ..., x_{l-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.

B.2 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.

B.3 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. Three constructions of such hash function are given in this annex. They are denoted as HBS2PF1, HBS2PF2, and HBS2PF3, respectively.

HBS2PF1 can be constructed as the full domain cryptographic hash function FDH1 in ISO/IEC 29150.

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 = BS2IP($H(m)$).

c) Return h mod p.

HBS2PF3 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 *plen* be the bit length of p.

c) Let *mlen* be the bit length of m.

d) Let $i = 0$.

e) Let $h = H(\text{I2BSP}(p, plen) \,||\, \text{I2BSP}(mlen, 128) \,||\, \text{I2BSP}(i, 128) \,||\, m)$.

f) Set z to the *plen* left most bits of h.

g) If BS2IP(z) < p, output BS2IP(z) and quit the procedure.

h) Increment i by 1. If $i < 2^{32}$ then go to Step e), otherwise return "Fail".

B.4 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 = HBS2PF(I2BSP(i, 32) || m).

d) Let P = 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".

Annex C
(informative)

Security guidelines for the anonymous signature mechanisms

C.1 Descriptions of mathematical assumptions

C.1.1 General

The following computational hardness assumptions underlie the security of the mechanisms specified in this part; namely, the strong RSA assumption,[13] the decisional Diffie-Hellman (DDH) assumption,[2] the strong Diffie-Hellman (SDH) assumption,[12] the Lysyanskaya-Rivest-Sahai-Wolf (LRSW) assumption,[18] and the static Diffie-Hellman (Static DH) assumption.[8] Table C.1 below summarizes which of these assumptions underlie the security of which of the mechanisms specified in this part.

Table C.1 — Mathematical assumptions used in the mechanisms

	Strong RSA	DDH	SDH	LRSW	Static DH
Mechanism 1	✓	✓			
Mechanism 2	✓	✓			✓
Mechanism 3		✓	✓		
Mechanism 4		✓		✓	✓
Mechanism 5	✓	✓			
Mechanism 6		✓	✓		
Mechanism 7		✓	✓		

C.1.2 The strong RSA assumption

The strong RSA assumption is the assumption that the following problem is hard to solve. Given a randomly chosen RSA modulus n and a random z in Z_n^*, find $e > 1$ and y in Z_n^* such that $y^e = z \pmod{n}$.

C.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 of order p and three elements $g^a, g^b, z \in G$, decide whether $z = g^{ab}$.

C.1.4 The strong Diffie-Hellman (SDH) assumption

The DDH assumption is the assumption that the following problem is hard to solve. Given a cyclic group G with generator g and of order p. Given $g, g^x, g^{x^2}, \ldots, g^{x^k} \in G$, generate a pair $(c, g^{1/(x+c)})$ where $c \in Z_p^*$.

C.1.5 The Lysyanskaya-Rivest-Sahai-Wolf (LRSW) assumption

Given a cyclic group G with generator g and of order p. Let g^x and g^y be given. Assume that an oracle can be called that answer queries s by a triple (a, a^{sy}, a^{x+sxy}), where a is a random group element of G. Let this oracle be called with the following queries $s_1, s_2, \ldots, s_m$. The LRSW assumption states that it is computationally infeasible to generate a quadruple $(t, b, b^{ty}, b^{x+txy})$, where $t \notin \{0, s_1, s_2, \ldots, s_m\}$ and $b \neq 1$.

C.1.6 The static Diffie-Hellman (static DH) assumption

The static DH assumption assumes the static DH problem is hard to solve. Let G be a cyclic group of order p. Given $g, h \in G$ such that $h = g^x$, the static DH problem is to compute x given access to a static DH oracle, in which for any input $r \in G$ the oracle outputs r^x.

The security strength of mechanisms 2 & 4 may be weakened due to this static DH assumption.[5] However the attack on the static DH problem is impractical due to large number of oracle queries to the principal signer.[5] One method to mitigate the attack is to choose ρ in mechanism 2 and p in mechanism 4 as safe prime. Another method to mitigate the attack is to limit the number of static DH oracle queries by the principal signer.

To avoid the static DH assumption in the security proof, mechanism 2 may be modified such that J_I in step b) of the membership issuing process and J in steps b) and c) of the signature process are computed by the principal signer instead of the assistant signer. Mechanism 4 may be modified such that J in step b) of the signature process is computed by the principal signer instead of the assistant signer.

C.2 Recommended choices of security parameters

Mechanisms 3, 4, 6, 7 all make use of a pairing function. Methods of generating pairing-friendly elliptic curves are given in ISO/IEC 15946-5. For 80-bit security strength, 160-bit MNT curve in Annex C.2 of ISO/IEC 15946-5 or 160-bit BN curve in Annex C.3 of ISO/IEC 15946-5 is recommended. For 112-bit security strength, 224-bit BN curve in Annex C.3 of ISO/IEC 15946-5 is recommended. For 128-bit bit security strength, 256-bit BN curve in Annex C.3 of ISO/IEC 15946-5 is recommended.

Mechanism 6 also uses a conventional elliptic curve. Methods of generating pseudo-random elliptic curves are given in ISO/IEC 15946-5 and examples of pseudo-random elliptic curves are given in Annex C.1 of ISO/IEC 15946-5.

The following security parameters are recommended:

— **Mechanism 1**: For 112-bit security strength, the following parameters are recommended: l_p (1024-bit), k (160-bit), l_x (160-bit), l_e (170-bit), l_E (420-bit), l_X (410-bit), $\varepsilon = 5/4$.

— **Mechanism 2**:

  — For 104-bit security strength, the following parameters are recommended: l_n (2048-bit), l_f (104-bit), l_e (368-bit), l'_e (120-bit), l_v (2536-bit), $l_\emptyset$ (80-bit), l_H (160-bit), l_r (80-bit), l_s (1024-bit), l_Γ (1632-bit), l_ρ (208-bit).

  — For 112-bit security strength, the following parameters are recommended: l_n (2048-bit), l_f (112-bit), l_e (544-bit), l'_e (128-bit), l_v (2720-bit), $l_\emptyset$ (128-bit), l_H (256-bit), l_r (128-bit), l_s (1024-bit), l_Γ (2048-bit), l_ρ (224-bit).

— **Mechanism 3**:

  — For 80-bit security strength, choose 160-bit pairing elliptic curve and choose 160-bit t and p.

  — For 112-bit security strength, choose 224-bit pairing elliptic curve and choose 224-bit t and p.

  — For 128-bit security strength, choose 256-bit pairing elliptic curve and choose 256-bit t and p.

— **Mechanism 4**:

  — For 80-bit security strength, choose 160-bit pairing elliptic curve and choose 160-bit t and p.

  — For 112-bit security strength, choose 224-bit pairing elliptic curve and choose 224-bit t and p.

  - For 128-bit security strength, choose 256-bit pairing elliptic curve and choose 256-bit t and p.

- **Mechanism 5**:
  - For 80-bit security strength, the following parameters are recommended: K_n (1024-bit), K (160-bit), K_c (160-bit), K_s (60-bit), K_e (504-bit), $K_{e'}$ (60-bit).
  - For 112-bit security strength, the following parameters are recommended: K_n (2048-bit), K (224-bit), K_c (224-bit), K_s (112-bit), K_e (736-bit), $K_{e'}$ (60-bit).
  - For 128-bit security strength, the following parameters are recommended: K_n (3076-bit), K (256-bit), K_c (256-bit), K_s (128-bit), K_e (832-bit), $K_{e'}$ (60-bit)
- **Mechanism 6**:
  - For 80-bit security strength, choose 160-bit pairing elliptic curve for G_1 and 160-bit elliptic curve for G_3, and choose 160-bit p.
  - For 112-bit security strength, choose 224-bit pairing elliptic curve for G_1 and 224-bit elliptic curve for G_3, and choose 224-bit p.
  - For 128-bit security strength, choose 256-bit pairing elliptic curve for G_1 and 256-bit elliptic curve for G_3, and choose 256-bit p.
- **Mechanism 7**:
  - For 80-bit security strength, choose 160-bit pairing elliptic curve and choose 160-bit p.
  - For 112-bit security strength, choose 224-bit pairing elliptic curve and choose 224-bit p.
  - For 128-bit security strength, choose 256-bit pairing elliptic curve and choose 256-bit p.

Annex D
(informative)

Comparison of revocation mechanisms

This annex provides a comparison of revocation mechanisms.

Part 1 of ISO/IEC 20008 introduces three different levels of revocation in an anonymous digital signature mechanism using a group public key, namely:

— The entire group is revoked.

— The membership of a certain group member is revoked. As a result, the revoked member is no longer authorized to create a group signature on behalf of the group. This is called global revocation.

— A signature verifier revokes the authorization for a group member to create a certain type of anonymous signature. After such a revocation, the member to whom the revocation applies might still be able to create other anonymous signatures on behalf of the group. This is called local revocation.

Part 1 of ISO/IEC 20008 introduces four types of revocation mechanisms uses revocation list according to the content of the list as follows:

— "Private key revocation", in which the group member private key of a revoked signer is specified in the revocation list, and a verifier can check whether or not a given signature was created using such a key. Such a list is usually used in global revocation, but can also be used for local revocation.

— "Membership credential revocation", in which the group membership credential of a revoked signer is listed in the revocation list, and a signer might be required to provide a proof that the membership credential of the signer is not in the list. Depending on the mechanism, such a list is usually used in global revocation.

— "Verifier blacklist revocation", in which a signature (or a partial signature) corresponding to a group signature linking base is listed in the revocation list, and a verifier can check whether or not a given signature was created by the signer that created the listed signature. Such a list is usually used in local revocation.

— "Signature revocation", in which a signature (or a partial signature) is listed in the revocation list, and a verifier can check whether or not a given signature, along with a piece of evidence provided by the signer, was created by the signer that created the listed signature. Depending on the mechanism, such a list can be used in either global or local revocation.

There are four revocation mechanisms specified in this part of ISO/IEC 20008. They are private key revocation, verifier blacklist revocation, signature revocation, and credential update. Credential update is a type of membership credential revocation, where each signer will update its credential so the proof that the membership credential of the singer is not in the list in inherited in signature generation. Table D.1 summarizes which mechanisms are global revocation and which are local revocation.

Table D.1 — Categorization of revocation mechanisms

	Private key revocation	Verifier blacklist revocation	Signature revocation	Credential update
Global Revocation	✓		✓	✓
Local Revocation	✓	✓	✓	

Security proofs for the private key and verifier blacklist revocations are given in the full paper.[4] The security proof of the signature revocation is given in the full paper of [7]. There is no single proof for the credential update process, as it is mechanism specific.

Table D.2 summarizes which revocation mechanisms are used in which mechanisms. Additional revocation options may be used other than the ones specified in this standard.

Table D.2 — Revocation options used in the anonymous signature mechanisms

	Private key revocation	Verifier blacklist revocation	Signature revocation	Credential update
Mechanism 1	✓	✓		
Mechanism 2	✓	✓		
Mechanism 3	✓	✓	✓	
Mechanism 4	✓	✓	✓	✓
Mechanism 5				✓
Mechanism 6				✓
Mechanism 7				✓

In comparison of revocation mechanisms, the following features and capabilities are considered.

— Whether it is a global revocation or local revocation.

— Whether membership credentials need to be updated.

— Whether a member needs to perform additional computation for signature creation.

— Whether a verifier needs to perform additional computation for the revocation check.

— Whether a member is revocable if its group member private key is unknown. Private key revocation works only if a member's group member private key has been published. If the group member private key is unknown, then the member cannot be revoked in the private key revocation. For the rest of the revocation mechanisms, there is no such limitation.

— How revoked signer is specified.

— Whether the revocation list is specific to a particular linking base. In the verifier blacklist revocation, the revocation list is specific to a linking base. If a group signature is created using a different linking base, then it cannot be revoked using this revocation list. For the private key revocation and signature revocation, if a member is revoked, no matter which linking base it uses, its signatures are revoked.

Table D.3 — Comparisons of the revocation mechanisms

	Private key	Verifier blacklist	Signature	Credential update
Global/Local	Global/Local	Local	Global/Local	Global
Update membership credential	No	No	No	Yes
Additional member computation	No	No	Yes	No
Additional verifier computation	Yes	Yes	Yes	No
Revocable if private key is unknown	No	Yes	Yes	Yes
How revoked signer is specified	Private Key	Signature	Signature	Credential
RL specific to a linking base	No	Yes	No	Not applicable

For private key revocation, if a group member private key is exposed, it can be revoked and placed in the revocation list of this type. All signatures generated by this key can be linked by any verifiers, including the signatures that were generated before the key was exposed. This means that unlinkability depends on the ability to protect the group member private key for the lifetime of the key. In certain scenarios, this revocation mechanism may discourage users from revoking their own keys when a compromise is suspected.

NOTE 1 In the private key revocation, a member can be revoked by the group membership issuer or any verifiers, but only if the member's group member private key is revealed to public.

NOTE 2 In Mechanisms 1-4, it may be possible for the holder of a revoked private key to be "framed" for signatures she did not create. If a malicious entity learns a group member private key from the revocation list, it may obtain a valid group membership credential on that key. In Mechanism 3 when the group membership issuing process is run by the signer with the issuer and in Mechanism 4, the malicious party can obtain a valid group membership credential by re-enrolling with this private key. In Mechanism 2, the issuer performs a revocation check during the issuing process that will prevent the malicious party from re-enrolling with this private key as long as the issuer has a current revocation list. In Mechanism 1, the original group membership credential can be observed by an eavesdropper during the group membership issuing process, unless there is a secure and authentic channel been set up between the group membership issuer and the member.

NOTE 3 In Mechanisms 1-4, the group membership issuer can create membership credentials on a group membership private key that he has learned. Therefore the issuer is required be trusted not to frame the members on revoked private keys.

Annex E
(informative)

Numerical examples

This annex provides numerical examples for each digital signature mechanism specified in this part of ISO/IEC 20008.

E.1 Mechanism 1

Security parameters:

l_p = 1024
k = 160
l_x = 160
l_e = 170
l_E = 420
l_X = 410
ε = 5/4

SHA-256 is used as the underlying hash function. The outputs will be truncated to keep their k least significant bits. HГ is realized as HL function in Annex B.2.

Group public key:

n =
72E3DD15 AA189A85 D8169EE3 78E2C310 773BDA0A 8C21813F A2309CBF BA0D8BD4
64CA8774 070EE0B9 B6726805 D9D61D2D A42D0347 8447C6FF 5D297116 B5F83211
C8FBB55A 89BEE236 A4856F26 13DE5531 27F8AD43 A1716E10 009346F6 31509055
D5148DE1 5C3E107F B3F14B7F A15D4F15 4E59ED89 34B38A78 31AD0997 1871689B
7364B08E 780EAED3 DF0326A1 204ABD56 EF9853AF 172FEC1D E53253CF 08C80B2C
A25BE0B0 7AB8F661 DB1CBDC8 5E0A4E80 ED0671D3 25182293 07FDFB8B F8F77D0E
33D5CBC1 C8A8A2AF C43261DD 2351E1BB 0E62D77B 9FB388CF C18EB5D6 36E23703
503ECF2D D837D202 02678B4D C408AF9E 5580433E 61168CB3 DD877C3F 4CA4EDED

a =
0F530B25 19C8AA74 B8E3E632 2E7F9617 8AB8BB3C 1FAFBC4B 0BAFA43D C2D45A9A
16DB7C46 F4477DD5 C74B4FA8 37838BD4 3D6500C5 8DE91529 2EBBF49D 6A16BDA1
889B75D0 C0E0383B C8F02881 F63A9144 DDCB2E39 424CEF64 CADB4D9B B08177B0
C04DC0AD B656C4F5 2F12AC8F A17C9FCC 4A1EBDF9 859B263B 7FC817BD BB5034EB
B84BD698 FB79FD7A 45008B1E 0843F34E 39839E95 304DC1F3 0A32952E 421D4288
FC5E1001 AB16A37F 3AA6988A 13B6F6C6 D9568F3E 28A38657 45393F68 C843622B
A878922B F6E5C309 36F93406 8DBA39E8 ECD30FB1 FF4E5EDC 6EC605ED DE227E3F
4D54501B 6CA69D5F 572E2F56 AB0EDC22 A87AB7C6 7543C8D6 E465EC19 C51276D2

a_0 =
65451CF2 6AA403AD E92A9F13 A34152C8 041A9247 DB362D42 A60E5DBE A07E050F
518737D9 8C16092B 47ECF4AB 075622A0 FDFC20A4 C4291174 2F76DF96 1D1E840E
548A06FE 859CF811 E7AE7290 35ADA222 44C27516 27781334 6EF043FD AA9D0844
8334240A 093B8DC6 0BFAF928 4AAF9177 BBBC49BA FD4D2830 F604C8DD 6C1A184D
5087730C 99108452 3C30CD2A 3AFEBA32 D72DA861 F0377CD9 5B07F52A 32ED0D28
3BF27F5C 3C3E4578 81E4BE7F 522F9522 F9670D3B C98AC2F2 291BEEEC 973A3479
D0A2D08E D4AEB51E BAAEC971 55F7E430 083955F8 A1610212 A2484FF8 74767056
27706C2A 877D1B13 25589D19 16681056 D5BE2A7B 502C2327 42873B48 07E4C1C1

g =
466402AF B01C8AE0 74BA7E6D 3EFF5CF1 AA754ACE F1B5FC45 C58ACCA1 4C84576A
FCC09362 5EAF074F 723F3BED 8E552E62 5C334C6B CA179580 3C018B4B 8B9329F8
1276DDDD 454E4318 6D8C4D9C 35C9DC07 70E1A1A0 EB27E5F2 17BAE6AD 1EE74712
BF21CA80 9FEA88BD 8D2F4B2A A9F50516 467AF41F 617F3DCF 8BEC54E0 B0DE32F0
D1DFFBCE 434FB1D0 A6789FC7 381704E2 82EC8859 95017AB6 BBD8D627 88AEF089

```
F973FC67 4192271D 3C64C750 07861325 579A63EC 60A7DBCF 23C921E5 B74CE2D3
6F824562 D405FC8C 797EF97D BB6C1E8A 7B1F57EA B0E4D063 23C5F88C ABD3A6C0
653BAFD1 155B6A69 37A62A07 9099FA11 6ACD068B EC7AE221 5010BFA0 AE861FC9

h =
13C4A935 45BD9A55 ABD12349 1F173DC1 75BBA355 58B5A9B4 31E22402 1EA2483D
1F9685EA 6552D0D7 8A4BA800 C12ED12F AC8458B7 B7FB2662 3C40244A 70710878
BE952AAC C3E35D93 C039EBEA 1F554473 C4904A9A A460C670 AACF7CE3 504C4916
ECFC08E2 528183D5 5DDC1AD1 EDCF645C 66ADD64B E6EF8953 F7690C41 E5643211
0982D6FE 27601EE8 8404081F 6010CB3D 034E467B F7356E48 51FCBA1B 566E3DAB
DCE9C1C8 E7276A23 04A0E066 44649BF8 CF938313 AE7F8494 F91BD3AF 2472EF3E
89B1D124 581002F0 B4124EF1 5AD5FB55 EFD78AF8 DCE73109 E4DE7504 EB458193
A828A8F1 022FED61 D70D02C6 F6C805B0 E0C71546 8D269CCD 8809A16F 4C54154C

b =
3FC0A1C7 C715150C 7D0DF108 DF8A0528 FE4EC7A1 48A066A3 F156C3E8 467291B3
55D47D60 00F010AB 6C341D90 C633870E 7B65BB4F 94BD7D07 9E80A8F4 55B8087B
0D82ADB8 3DCB292B 79B59B8C 812341B2 593FD3B4 1D3A7A44 983BEAFC 8C97EB02
FAF9B756 B188F365 9E65D07F 380E989D 658492BB 7AA5B7B4 141F36B8 090B0E0A
6E8FD4E2 B24947C3 13FE7307 94A4CB36 E326909D E66FD924 84DB4A71 15DDC5D0
9B1FD469 479EEE71 CF2B3313 D8208166 F2C15F4E F2DCCA79 3B7F8D27 4F5B5D85
735E04B6 A0A3551C 9BA6EE02 496C8035 D00211E1 58F86460 C07199D4 8608C4A2
58D937ED AEE23E51 7DF14162 72F5C72C 3EDCD55D 959ED4DA 05759D24 E3BE5823
```

Group member signature key:

```
A =
375B6697 828C1CB3 6FF9E3D9 7D7EA508 D34E09AF 40ACAC31 70757009 386A6613
8D76A861 1A0C9FED 306471C0 FBD0CE41 277AE81B 42F7CA20 CB831C99 F54E4DB1
305620E7 93E8AF3A EBD922DE 45FBC66D C5DB0A70 8E7001B3 3497CEB2 D74C5652
091977B9 C4CF5356 941CDFA5 A37FF2D5 807262FA B2284DAB 3B5E68AC A927882D
E4A4882F 48F999ED C49B38F8 D42A7CF8 1955E01F 4884CB69 40D746B6 A25ECD98
D61D351F 0D2B6A5F E9D3BD81 6521D350 AA47D3AE 6CC31A9D B354D300 5E143BEE
34C8B7D2 9381681E DA5D68A2 EA63AD4A A29CE58C D9F2B690 5B42CA9F 2651DFD6
D2E0AD8F 91E3A1B0 B9ACCF25 5C47BEE0 63306943 045373E3 B4FB53E0 5149C84D

e =
0000000F FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
FFFFFCEB 723BB231 7D993CAD 5EE5F0C2 4F6BD30D 5446A56F

x =
04000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
3E75531E 1AC338DD 9B150FBB F9B2F984 40A30BDF
```

Signature process:

```
bsn =
00313233 34353637 38393130

Message m =
"abcdefg"

f =
453FD0DD FDBA8F31 2A1CE338 F2767D9A BCD031FF C42C45B2 B0328B9D E410BC1F
C23D3FF8 AC3807C6 52ABACAC A579005F ADF45BF4 3D208C0E CB717317 4F4D2AC6
F9313E2E 41EE1F3D AF3264E4 89848B11 A5570E97 5BEFD851 5127FD0D CA78FB62
7000EF26 64AD94C1 802D0ABB 3CF81E74 FCAF9930 DB1DB12F 3805F1D4 A540F706
34DC2953 0D09DA60 D3337D20 61F6388C D0D49087 2B9C9D0E 2F2CFD04 BD167D28
F84E0BDA F3227C69 52586FF4 FF2418C1 67B067E5 55D4A77E FB3ECA4E 58384605
5C8FEDC9 BD3F3E4F DFFE6F95 FF4E8199 73D54ECB AE183533 6B4BDFEF F9F3FA2F
F677450F 7247D2E9 EA74200F 325492FB 14BF85A0 7C249E54 7635BBDC E9ABDF77

w1 =
903FF56B C8865FCB 058A162F F71734F5 6AA67822 CFFDFBBF 266D6EAD 3B81155A
4E317776 D9B2B351 59CD0C9E D8A3766B 9781EF3C 8CE19EBD 0DD2C5DD 45D5816D
6D5085A2 060CC78E 1D098071 7DF8D934 D439E460 1A360FFE 88C3841B C4D98091
CAB4726B 8691FCF6 FAABC40F 5938A9FE 01E7A93B A77AFEAA 29BB2D9E F7BCF534
BD895888 E1DEE337 9A647719 7D70944C 7F3A13E3 44F88E81 639DAC44 128439FA
```

```
79C05588 45375525 00C1D51D 56143961 740E394F 61C2DCED CE468A85 CB869825
95B80F52 F966A15B 4DEA4D12 359B8FAB D9277AD6 561974B7 2BB8D7E6 2785DC5F
FF502DEF 7B1630BD 4A430084 CE13C685 0432A943 DDB1B767 2F1EA8E3 4E74B149
```

w_2 =

```
3C596556 AA7E429B 1703FE21 BEE7D260 3092AD99 12FD90EA 6671B8EA A901BBA1
C5F2E3A0 0EA6CDAC 7B5A9E0E E3748AC3 09F69D51 D8B7F423 2D5237EA 34D6FC21
9EE94387 1370974B 9B6C8E49 1E9A0C92 FF4D1C51 759891F0 1E5B9A5A 04AA5B64
4A228412 056C1D05 95F06020 8570587F 96D49E98 665CA6D3 866F8249 2FA2E1E9
0C87A891 6E15BF72 927C4446 E53E0F5A 09FD6D16 158B9016 0A726C13 9214A0D4
DF2FEAF6 AE223430 D97424C0 20209D49 131169D5 10EC35E6 41AEF4B7 B2B0CE8F
A8C73910 AD220D0A 168CD86F 8EA09187 12121EC6 4FA286AB E9DBA4C1 B1215D58
6E5F4E40 F9FFDAC3 F51AFBBE B1320B28 824F53B0 DE79FE63 9CD66590 CD56F2A6
```

w_3 =

```
2763853E 200092FB 842CC54D A92B377B 4E2FD10F DEE0FB24 E5718D59 7AD0EF99
8023E2D1 424F817B 1ECF7017 C470878D 8FB5F2E8 694D11F6 631F075E 5B2617DA
99A8BCD6 BD6281D3 E86C58DC B4031BD8 A5756F4C 20794583 C1F61D83 FD81E025
7F160843 B38B4DD5 CE8423C2 3DBDD7BB C2CF698A 78D72132 442D3A83 D963D3DD
87538F97 3EBE1D0B B147C9F3 FAC8EE84 8D25EA64 004F1254 D325844B CBBE8424
06FE0120 A5E14703 9B246776 04875F24 9433CBE8 612A06D4 B817B0EA 66C8F3F7
80A35798 0425CC2A 8CE6A6A4 24BCFEF3 8F7F0697 E7A2ED23 6D59EB3D FFFCB69B
92EF74D2 33909BB1 31709001 1AE931AE 6E8C1A18 D96AAE32 BDF504CB E2477570C
```

T_1 =

```
5C96F1C2 EDA116B5 6C5A28C3 8098950B DE084521 6E1A5FA2 0F985999 064FCCC1
9946B54C 9A21F41B D5713079 16C8BF52 10FFD01B C3C0BA65 09B59C87 68B17137
22845000 4843DCDE 201F9C01 C1A48948 A165A2E4 13CBC416 F6577721B 4AB69F3D
92EC3039 E0E82620 03F69444 275C36FD 20A8DDB7 4BE29928 66A2DD7A 8663837E
05DD224A 8DA92BF4 1CD5F9F5 31DF5DCB 3112F0C4 229D7DDF B0FA13E4 5CB5B794
DCB40C2A 3E42D593 E13AD04F 8857D361 58BCE8E2 64C07A18 0C81AD49 B8EA3790
DE52B00F 5B403AA3 26307A79 29AEB630 53183B2F 5946AB34 B45CB1C7 AE76F402
FE9D7396 FA3A140F ABCF31A4 A6D1BE6C 55B89D4A 6B3A4DC1 51242787 56E7A4AF
```

T_2 =

```
44F0115C EFF930AC 803726E6 614B46F9 2031OBD1 68956D60 3E771237 7B512D02
CF04A986 943BB3F8 5F4C4D12 4438AF50 A1AD13C6 D7FF5666 74F2241F 6386CA60
55DF0BFB F5860275 C450CA14 FAD2D40B D97245F5 B029A009 AD7CFF15 66C57517
15591D64 93473277 9EA974C2 E21B3467 E42AA151 5BE575B6 686A666F B9ADD370
316007B3 312887E3 7BDDA993 049A00C0 C884AA55 FE752E20 BFB6CE7A 9F12F5A2
3D9053EC A7EB8823 21BC9B19 BFE59B4C D3BD32C0 C12A52A0 FD3C6B02 1386A207
1286E1AB 4614889C 941EE9D5 13A7B3C7 32B0718B 4BBFE878 A2793DAD C561A1B0
3AA80A55 80004C58 1D390C69 2577875A A0324F75 05A89AE4 7AB61455 1F0B0550
```

T_3 =

```
1910180E F5ED5B8B 166E43F2 CABBBBA9 038FCFA2 DBE9556A 54E8D5F8 523577DD
5AC89BBE 510809A3 B4B2A6B8 F8AC3A0E 9AE6B964 5EA1B41C 5E2577A7 0D080FE9
675DAED6 65796AC6 46CBC74B B4B2D427 A805C096 E4DEFAFB 707B4EC7 E3965EBE
7907F09F DEF70AD6 1CF8CBD4 39B3AD69 76EFDB7A E36C6CC0 1BACD171 090392C7
86453340 3A4AB456 A2E16DD9 B84A6C41 24870E87 DDC8578E 72E055CF DE15F7FB
C4A7000B C9CF666F 9B5877E4 8117708E 8B5071FB A2BBFE4F ED8D346D 25A19FBE
609D067C E946CD56 C6B4A81B 4A8089EA 90904077 4792631C B868E450 70427A48
16B6832E ACF5C5FF 864C4F3E 6B6F78D7 3683E379 B1186EAE C9C61F2D 7E6FB0C7
```

T_4 =

```
2CD4FD7D 0B491E59 C73B5083 B64BA089 176B5C46 A20FB06D E1B104B6 1F248CB1
10F7C259 8C153A0C 8C9D5281 480B10CA 0C587174 573F70B2 C54B3C91 28CB0AE8
4155A699 D29C5028 796744C3 685AD22E 4425E2E8 9789DB76 FA1C0119 12817997
2460EDF1 0DC6B484 5F3CD951 A8348BEB 9C88FA17 579A710A D66C7E90 BA84B724
A3645C41 5DAC6939 0BB73E83 9B9ADFC3 F6118304 3BD249AF 4A14C058 8EE8B0D7
59487B11 D35EB9F0 DFEC0636 925EF3B3 BFE3CE98 72CF73CD DE2EEC2A A251B984
E206043F 9061F375 25491E18 3295C134 5500997D 7D74F92E ABA5915B BA993D12
6F9756F4 C8874899 F58D6F25 E28640A9 C3A8AE11 F1E56338 62C62308 4ADDD1FD
```

r_1 =

```
0897F8E4 9394A6E1 BC625487 A21187A2 DA10383A 2AC272BC D80CDD35 A346363C
     672CEF42 7A69C199 9E2F998E E40C3D39 3A250520
```

r_2 =

```
000080FC 06D07109 8A6F7870 6F973875 EB851D99 15C3B5BB 656DFA7F 395066DB
8AE2223A B1D2E6C2 86778413 78F2F662 E3A694D6
```

r_3 =
00000069 63C9FDDB E0BD59A0 6BE8CE73 3BEE8F55 EEAC8C12 63586346 F5628510
89C4C6D1 B323410C 350D8F1D 65433BC8 F5090639 9C57EE0F F5537A7C 510F1BED
C854DFE2 91970C9B F40A7D3A 0465BAE5 0A290176 1B935C5A D88641E6 DFDFF3DA
929BD6A9 AF1D7E28 D04D6676 32695AF6 EFE4F0CD 29C9394A A5594E0F 6C5292F7
D3085E44 AD9FDCD1 4F0FCB5A F7C15B1F F917E39A B685E634 6B3C8A58 2C65AF39
9201F20F C5286DB0 5CE5E069 BAD92CBE E80CFC35 8B0716AA 93334E71 C622EA15
DE2D4E90 39B4F2E6 52D2E942 7CD5FE33 B2139AA3 0A4FA342 4F575CB7 08F32014
E42083B0 C5FF3E7E A1FE46C7 3E22C48F B1AE2472 E7D5A375 4E6873A3 098FBA71
D24F7E5B BAFCDC9B 5D092619 326BDB7B 7B9D9EDD 3FA7D6BC F4BA414F BF3AB03C
E1D9C5B1 4BB6E2EE 9A39EFDB 9CB67682 EDD9574B 452F5C7F D40FD5AD 329C980B
7B4BF5AC FCB933C1 E5C5CF7C EE325304 DCE83663 563BE323 CC8BD21C

r_4 =
0000003C 8FB37C34 73F651E0 1300DB3F 61BC5162 70FC2E62 417B13EE 07AB3813
4F458F4D C6B5163A B1FDEF00 8252A446 6B2E2D25 3678F22F B9034B1C 158AB04C
F47A72EA 4ED5F469 77BBC48E 1EBB6906 00BE40DB 460F0B27 A6A5FBCB 4648B7E1
8C9A03A7 F329EF53 9070AA9F 74CB2735 07A0A660 42C5A328 E829B282 31768FEF
7085973B 580F68B9 52B4B02B 8F8A894A 50277382 0E140885 5DCD5C53 84E401AD
C471E646 D02A1AF7 9366C86D 50655EE1 211AF90A 4FA2454A 6FB49379 0519E780
E9607154 33743550 43285DD8 E37F247F E3B4DF11 573E4C91 2B1677A0 14E71A45
0A8690EB D1836D9A 7DE9A132 26B9E50F 562F64E5 46233DF2 7CE4AD46 7A7884F0
E7641617 00B38DC3 A1A714CA B99EF98A 8D9D330F 37A81605 F33A5942 6BF4D16B
67734E1A C3C2F7C1 41DD9982 8BEFFC70 977A026D 9FAF50BB 2AFD9BC1 227E652C
6D9AC4DF 3C1A4DBC 8B465C82 205653DE AF2F1408 B620BE84 29BB887B

r_5 =
0000003C 9621E1F7 CA5DE96C DCA4AEA4 C1737836 31E69FA4 B043B89E AF00749D
B6FEF425 ED3CB308 E5888F5D D466CCC3 82A7BB7D 8888ECEB 939324CF D93B1AB2
637AEC07 82D38F1E AA92033A 04D754E3 2501B00F 94DE953B A920F18D EBCE7B35
F916A8C7 05FA2A06 C0E1F8B4 113AFF82 082B7A8C 167C7CF8 7CC7A6DF AC908E2C
7C39AE54 8E3DF467 57C97E60 6443AA60 C286BFBD ADA8BDC7 658929D7 83328942
ADEDB7C0 4D85E20D 72066918 FF01D3E0 4994C0A5 387DCDAC 7AEA530A 6999C798
8AD99B43 7EB97788 8E5DA30C BA911F81 30216D42 E294C33A AB097285 7AE86A06
4A3A42D3 D9839BAC F473617F BEDFF35E 14546C4E 4098EC16 5258A309 585DDC7A
632E254C 6C2CC0E0 C3760A10 01BA6A9D D9DAE9EA 74BE20EC 49AB2568 B4DBC913
6A8E87BE BB286EDA 595BF8D9 2C386565 172FDBBA 492EDECA 01C08C16 2140D08B
F0D2235F C566365D E643A9D7 9ADFBE93 B0BCDCC0 99B48F66 4398E5FB

r_9 =
08EE4D7F CB83E628 8FFFE433 7A4E6501 183249CC B3B5652F F5F80E11 EE67A097
60BB53F7 836C86E4 99FC68F8 D7CA04B8 8507BCF1 9402591E 6F39177C 8B2BA0A9
64A22088 25847FE7 644C5E3B DAD186B9 E9769DAA F4FFA74F DDB2D3EE 30913A7F
5F194D3C 38E62D4E C47D0D15 37BD1CE5 A51EB01A 6F7F8095 196B9E2C 84E73422
2DC41C5F CDBE2ECF 144BAB4D 98A29E08 AD551BD7 E3B6BC53 896DFA08 AC56D11D
17C88766 5E51D73D 63C78E17 40038927B 1B54138E FAC0EC23 1DFCF6CB DB8999EE
85F7EA55 384543E8 FABDE89F 1A28F6B2 FAF07BE3 690FB798 BB84EAEE D86E3909
7D4506D7 1C3ECAD4 BFFE4A7E 048A7AFA D874A576 DA3D87F1 DD853375 504EC9DD
326C148B E2383202 58CE6988 2357E719 37F77D1F 6B565D11 F272F222 1864CB97
1A10F637 A10BEE84 3A945447 42675D1B 8482B282 B2610E4C 194AC612 A324E539
76BD228F EA17898F 5098BD20 496BD782 E937FC79 C51617E9 A175E20E 049ECAF0
9F6318B4 89E9F10C 51FCF1B6 0EDAC6F2 091BFAA7

r_{10} =
00EE6A20 56206A63 78E71D92 A4D96CAB C820D0BD 0B43A5A3 9E9CB7C9 681B92E5
BC4187C4 8A7092E3 20F0D654 DCF49D57 08A55292 A2BD27BA A7FAC208 234CB3D0
F6436D72 2DE31ACB 66410105 D688454C 6D2BDD33 AB3C98BC E53363E8 F52EC55C
E3ED40EB FB145B7C 63F7C5DD 74293B51 4DB2010A 55EF1AE3 FA319866 ED327FF8
E0A9A05A 07B446B1 50823FA4 B26F9DE4 1F1A4FEA 035B0675 FD7408FC 129106A5
2B81DC03 5A844549 0071D608 81F99946 FB864E56 FFBBE04C 216E9B2E 7ADD5791
A40751CF 0E6D8FBE E296FCAD 5D7DEBD6 F3CCC79A 4DC93242 A540583B 4CFDD854
565061C5 B4DB9FDD B336F735 05AAD762 B8230813 D09E21C7 20561690 1C3F4D6E
C89EF8CB D0E22EBF 28E00511 3B95172D 9E66BD60 97545BB8 B04A1442 D25FF5D5
7AAC0350 4980AA5A 41F62DD1 7EE4B3A0 69F04861 EA51D8B0 9D83DC8A 53C251AC
FE7D30B8 6D98D3E2 53A7AB48 F4FDD5CE 044F541E 4C774873 4FFEDDBD 13E94D98
9B9CE959 A27CCAF0 2692AC8E A80C59CA D6FCDBD2

d_1 =
047E2731 97415CF2 5F43729B 0DC37A27 24EDBBA4 335B4ABF A56F5F6E 9E056590
139E0310 662D903A 939DDCE9 5D24FB44 B2AABAAB 47D2A2DB 1299AC33 49BA1FDD

```
31C7C571 5EB626FB 0E127168 355B1CE0 2964DC5D B86D27E9 70BB21DC 815BECAF
493BC724 EEA46539 C82CFEA5 71FAFE42 B0A64782 0600AFD0 5FE5ED0C 550545A7
213BB3A1 F08BF173 FB533BBC 4E28C23D 313F4EC7 0E636B09 C190546D 37D6126E
D4764E97 7E7510BC 0D9D3A0B 12C36B58 5D1C3786 001DCDDB 08A4F5BF B048CBFD
36D675B5 66AB4903 9A74D0BD 8049C5D8 A00B4254 22C80554 91FEFA3C 6BD9F690
ACF4D989 12B66E19 970A4A69 BB018268 452107F6 E2C5DBB2 CD3E079D CB49DAB8
```

d_2 =

```
2E832286 06A065EF 321A4544 B83219AC DAEBC746 CC4EE586 86014626 329CC81F
945616AD 7D7D4C09 712EB9AC DD63C34D 832B1BF1 2663BED2 3507F7E8 5F3F3E62
558064F5 C4230B33 F2774F2E 79BDD0AF A35944D1 964C5322 14F97B97 D00D8554
D41133F7 44C4CA80 B24562F2 73EC44B4 3A4884B3 6D04CCEB 1E43D619 1228684C
28483E08 5F488AA9 1971576C 26867706 53853295 403AF145 34BA1113 2DDEA273
B4E12435 9670753F 688A7921 32260700 97D98440 38A3F6C3 C20D0C1D 49C9BAD9
EAA52AB3 216DD002 DB8FFB88 7547D0CB C8D1DEE9 B5F4AB17 85F12519 E30ED49F
68E0EE44 CF595DE2 3904D788 F9422497 1B98D3D7 38E7AFD0 336A368A F7CAD447
```

d_3 =

```
5ADD5BB3 018836B6 37F20D00 EA396C0D 6AB0F07F 01026D36 BCA1BDB2 CC600C71
B1565B13 1537A8C5 3DA7998B 99234628 77BF3D27 8FA21AFC BD3D9177 4DCA8485
6C35547B C021A0F4 571EB589 4FDC9B0C 71EA4844 C5202753 38208D2B DB5AF904
F47C3E69 98003EC4 37D84037 ED624AC4 4487781C 9E6362A0 23ABA6ED 1A6FCBC4
5FA6AA66 B58A51F2 61FFC110 72532411 6A5F61C9 88C5072A 71BE1361 A91A1368
1E7BF74D C6178691 88385F53 91D6E02C 71D58C28 9746ED9A 1D6A62D8 329B7518
85E13652 C1AC61CE 7E2C09D0 42EAD4D7 5035FAF9 D2A10865 1074EBEF 169E228A
35EA55A0 676D8DBA 8CABB721 16C1E6C9 2360E361 E674DC1C 000E0A4D 6DBB1E24
```

d_4 =

```
41C114FA FF5C1358 61E6DB82 32C9D368 5B1FF404 C9273A68 16A206FE 77E33E4B
E83A740C 48D9FBE7 AE50FB52 661BB60A 67D8E512 0F500CE0 FCB98A0A F3A05B99
24BCBB4E E655E426 3827FB1B 98C51D92 ABD961D1 D117F248 B0E078CA 7E40CCA9
E4BB0EBE E5E197E3 97B62D1B 4778B2A3 7E4973D7 B17C91FE 68054367 77637376
764E9336 7B451F7F 01C67108 77380C06 3701AA59 58144F2A B604FFE3 1B450C3A
C2C1DF92 B101B1FD 21F78E15 AE7F7C4A 33B3815F ACD3D2EE 31C4714F 0B79F594
862B5C9A 11BD89F8 94ACD8DB BB310CFF 203CF0E5 41F0E2D6 138E0B6B EBE627B2
471098DA 388F71C1 46FD4B03 5EDCED94 700BBFEA 79EB30EB 4313E82C F6877C83
```

d_5 =

```
4170D919 99B8BD2C 28CFC6C9 6682C0D4 570A2DE3 46734216 10F5F5E0 B8FE0C73
55B1A758 4162D5BF 3CA97EF1 C9E58B0F 59C2AC71 D293DC4B B79B80F2 53322535
8C93D00F 152004A7 9DA6B065 9096A5A7 12650702 E5571C2B 0A106CF8 7271B0D6
ACBCECCE 4463ADBC 1FD2A484 E91C4AB6 03F1507A 6B820C25 84CB3AE7 765194F2
2C1EDD8D E3A0BA35 449CD004 A3EC30EE F45230EB CECFAFA1 1FB8E637 FB9C2C75
C0B24140 C2A6CE20 BF9B569D 583C1C01 1F821EFA 16D3B792 FD33F388 E66BAC54
92FFA95D B2AC0CA2 5582FFBF 565C8F1C A9499107 59A6E101 9894BB57 C1A5A030
6CEE6326 EB35193E B245A3AE 829F6929 FB5EF611 0B84C4A6 03316161 3F998558
```

c =

```
44F1BC8E 392071C2 AECCF97D 770758E9 15941091
```

s_1 =

```
0897F8E4 9394A6E1 BC62555C 0057F04D B7A3EF48 04794086 3C2F8F79 6BEF2016
C2015954 B6A05135 682FBBEE 9E929EF7 549E6141
```

s_2 =

```
000080FC 06D07109 8A6F7870 5EC513ED D103D06E 18EEFE5D D6476614 439C5233
E76061BD 43FD44D8 9634FF9B E4A52A3C EEA2EB87
```

s_3 =

```
00000069 63C9FDDB E0BD59A0 6BE8CE73 3BEE8F55 EEAC8C12 63586346 F5628510
89C4C6D1 B323410C 350D8F1D 65433BC8 F5090639 9C57EE0F F5537A7C 510F1BED
C854DFE2 91970C9B CD314994 3CC47429 6A435E36 CFFABD17 05B76F66 F30C7312
308D384A B9BF2F70 0CB36A28 003BD1E7 FDE5E58F 255B8AC9 5BFE15B8 C846CB15
4B1CB4C6 1C4264E0 AA637536 9C168394 B333CCB9 8A14F16C EB6ADD74 5F7460A6
92F73E49 A48F4BB7 1FBC8FFB C0ED23DA 05BA14BC 5F860AB7 191536F5 6A73B95F
3C76E954 2012C4E5 07FA95B5 572A7A16 0B1CC6A5 4395C6BC 4AB4AF75 74796CCE
917D29DF C7AD4277 6151638F B222F7F3 914E6DC6 E19C4F0D 742F6E7D 53A7E2EA
E7B6AA16 73018392 B8F2D1A7 03272A5C B04DB84B 97742D73 6C298CFD 62CB1CDD
9412668A 27E262C4 B3126854 2AA44B3D D163E592 1EA3BDFC 905B017C 0CA856DF
7B1A8828 3735F33D FDFE0DBE D478915F 49085152 2551A40A 962AD7C3
```

s_4 =
0000003C 8FB37C34 73F651E0 1300DB3F 61BC5162 70FC2E62 417B13EE 07AB3813
4F458F4D C6B5163A B1FDEF00 8252A446 6B2E2D25 3678F22F B9034B1C 158AB04C
F47A72EA 4ED5F469 677B0907 8561F5A2 7E332130 63EC8ECE 269BF616 4A52671B
6A2445E2 EE0E714A 5FA0B8DE B708A11E E4AA1B99 944485E3 C6A7EA17 30CCB765
AA0C0695 CC0E1FC1 5884679B A55F4689 90CE5CB0 9CE361A2 509FA781 8B987E5C
0E7D6177 2AC8349B 4FA5DA7C 92B7BCB4 9E88CB6F 0DD69CCE 51616DF6 0EED1772
5D36C1FD D2DF3D20 60E55CAF 6391ED4E 6A3E9432 B814C504 6A5D0F03 ACB76B26
BAA1A765 9C9AE81E 9092BE0C 2B677B5F C54795E4 3B7C82B6 8CA85D66 54E32B05
D2EA8898 433529CD A7887C9F 9E232773 FF516588 4287F888 1BDBC333 2E061F3A
DF0338F1 9AADD03C 61006A01 F3ADBB1B 85F9318B 0DFDD509 A46D4140 4BC3B6EB
3C2DFB4A 497775A4 7D19B03D 0863334D 408650ED 4DB36968 8659B875

s_5 =
0000003C 9621E1F7 CA5DE96C DCA4AEA4 C1737836 31E69FA4 B043B89E AF00749D
B6FEF425 ED3CB308 E5888F5D D466CCC3 82A7BB7D 8888ECEB 939324CF D93B1AB2
637AEC07 82D38F1E 9FF66222 1D035EBB 6BD76CAA A125B4DE 5D62DCDC BB4EC542
427DD4DD E1DD1DC0 FFF67DB7 833FAA46 136AAF5F CEB08AC7 B66CD78A 126F21E6
3CFF6E71 0873B45E 169E548A ADC20832 5E90DCB7 8145A73B F2CBD66A A4B94186
0C2048D2 3EA47913 998A8BA3 64656CAC B5D2CE9F 548EBCF7 7D611286 265892E0
CB47DCA1 ED2B0B91 60A331CF EB94C1BC EBD37736 98B59381 DAB2B403 ECFAA9BE
CEB9836D 820A1F93 FDDC31E3 9D02FA1B D0E74BAD 66111F67 91087BC4 2B8C23A2
F05BB100 9D5D66D6 D2F288E5 5A59C676 122458C7 3206EEAD 9FDE6924 17ED6869
11EA294B 6354D7DA 16D4CC4A 46D94E2E 131BA09E 166CBD72 EB227986 0351D0D7
1456C4B9 7619D2E2 A5682C5C 36772AC9 96082178 C4031478 54CFD82F

s_9 =
08EE4D7F CB83E628 8FFFE433 7A4E6501 183249CC B3B5652F F5F80E11 EE67A097
60BB53F7 836C86E4 99FC68F6 6A36CA5C 0AF35137 95A82529 B5AF233F 5E3E78AA
976A1402 049A99F8 0E677327 4B4827AD C806C28E 68788343 B7CD7791 C6FAE726
AB02551D 2AEC4D5D 9428B328 642BAE70 FCB083A6 C38F3C22 BBC7A2B9 5A6F4DBF
87C70B3D 2695A262 033092D9 491BE4C5 CC3CEA05 2FDDDA73 4FCC87EA 0463B8BA
7E16CA28 8CC6C87F B4548FF6 0D2E62D8 25C5454C C08129BE 65F96813 38282808
3588F504 9E1BAE7D 7B9B018B F3DFDE8E A885E251 1133FE4 0A8306E3 095208A6
7C5054FF F006C18D F08E891E DF1CEF73 2406A8C9 541D9E2F 7FF48214 992F7F73
C854900E 73BEBEB0 E53B4F52 84DB4105 A8BD4619 4A3F980E 906894B8 5684781A
81376C58 E85EA4CD A529C378 A6FD3ACC 94E186F0 DE184555 587E3E74 A9554511
7FD2062E 299652A9 49E8589D 77EF42D0 F170D665 19669936 04C617AC C854023A
03B3E6B5 234D3935 49F1DD00 57E668C8 4B5E1110

s_{10} =
00EE6A20 56206A63 78E71D92 A4D96CAB C820D0BD 0B43A5A3 9E9CB7C9 681B92E5
BC4187C4 8A7092E3 20F0D653 D8E8E4ED 730E1C5A 7A0B2D0C 85D2FC70 22AC5881
36DE6110 06873E7B 148920A4 D9CA8139 E2A41F04 64263D41 7C5AC822 33A5583E
C79E5376 EB592D7B 730E4517 3511162FB BC6C6562 028A41F6 E404B49A 41D98969
743C7F68 8B7934C8 2973D044 7AF3649C F7A4F386 7B548A55 E427D3EA D64082E7
50C9DE64 D09FF093 098A4483 58F240C5 D56CB8CC 42ECA051 82D11A56 5D388DA3
1FF61E39 4FCF402C CFC23B3A C81B8567 01FFD8F2 0D1EFEE6 DAC56B7F 3DAABF4D
7F0415E7 C2EFA593 108EE47 9D7D1070 DAC9A75F 32E7BF40 D5983FA7 CE853612
16432BE0 6EEFF996 9F97205E A887BAB1 14C35619 90E4A91A 46FFE07A 6E409448
5530B621 2E4EE66F 7D3D6C43 E3943C00 A0441F48 86C5CFAE 63E5D5E6 873AE2ED
5DD79A03 1A997D5 33DB67B1 0A4EFB65 7695B1AA D34CFDE7 83B5ABF1 8522D526
185872D8 4149468A C4D12A19 99D85B32 1ADBCB38

Signature is: (c, s_1, s_2, s_3, s_4, s_5, s_9, s_{10}, T_1, T_2, T_3, T_4)

Signature verification:

t_1 =
047E2731 97415CF2 5F43729B 0DC37A27 24EDBBA4 335B4ABF A56F5F6E 9E056590
139E0310 662D903A 939DDCE9 5D24FB44 B2AABAAB 47D2A2DB 1299AC33 49BA1FDD
31C7C571 5EB626FB 0E127168 355B1CE0 2964DC5D B86D27E9 70BB21DC 815BECAF
493BC724 EEA46539 C82CFEA5 71FAFE42 B0A64782 0600AFD0 5FE5ED0C 550545A7
213BB3A1 F08BF173 FB533BBC 4E28C23D 313F4EC7 0E636B09 C190546D 37D6126E
D4764E97 7E7510BC 0D9D3A0B 12C36B58 5D1C3786 001DCDDB 08A4F5BF B048CBFD
36D675B5 66AB4903 9A74D0BD 8049C5D8 A00B4254 22C80554 91FEFA3C 6BD9F690
ACF4D989 12B66E19 970A4A69 BB018268 452107F6 E2C5DBB2 CD3E079D CB49DAB8

t_2 =
2E832286 06A065EF 321A4544 B83219AC DAEBC746 CC4EE586 86014626 329CC81F

```
945616AD 7D7D4C09 712EB9AC DD63C34D 832B1BF1 2663BED2 3507F7E8 5F3F3E62
558064F5 C4230B33 F2774F2E 79BDD0AF A35944D1 964C5322 14F97B97 D00D8554
D41133F7 44C4CA80 B24562F2 73EC44B4 3A4884B3 6D04CCEB 1E43D619 1228684C
28483E08 5F488AA9 1971576C 26867706 53853295 403AF145 34BA1113 2DDEA273
B4E12435 9670753F 688A7921 32260700 97D98440 38A3F6C3 C20D0C1D 49C9BAD9
EAA52AB3 216DD002 DB8FFB88 7547D0CB C8D1DEE9 B5F4AB17 85F12519 E30ED49F
68E0EE44 CF595DE2 3904D788 F9422497 1B98D3D7 38E7AFD0 336A368A F7CAD447
```

t_3 =

```
5ADD5BB3 018836B6 37F20D00 EA396C0D 6AB0F07F 01026D36 BCA1BDB2 CC600C71
B1565B13 1537A8C5 3DA7998B 99234628 77BF3D27 8FA21AFC BD3D9177 4DCA8485
6C35547B C021A0F4 571EB589 4FDC9B0C 71EA4844 C5202753 38208D2B DB5AF904
F47C3E69 98003EC4 37D84037 ED624AC4 4487781C 9E6362A0 23ABA6ED 1A6FCBC4
5FA6AA66 B58A51F2 61FFC110 72532411 6A5F61C9 88C5072A 71BE1361 A91A1368
1E7BF74D C6178691 88385F53 91D6E02C 71D58C28 9746ED9A 1D6A62D8 529B7518
85E13652 C1AC61CE 7E2C09D0 42EAD4D7 5035FAF9 D2A10865 1074F3FE 169E228A
35EA55A0 676D8DBA 8CABB721 16C1E6C9 2360E361 E674DC1C 000E0A4D 6DBB1E24
```

t_4 =

```
41C114FA FF5C1358 61E6DB82 32C9D368 5B1FF404 C9273A68 16A206FE 77E33E4B
E83A740C 48D9FBE7 AE50FB52 661BB60A 67D8E512 0F500CE0 FCB98A0A F3A05B99
24BCBB4E E655E426 3827FB1B 98C51D92 ABD961D1 D117F248 B0E078CA 7E40CCA9
E4BB0EBE E5E197E3 97B62D1B 4778B2A3 7E4973D7 B17C91FE 68054367 77637376
764E9336 7B451F7F 01C67108 77380C06 3701AA59 58144F2A B604FFF3 1B450C3A
C2C1DF92 B101B1FD 21F78E15 AE7F7C4A 33B38157 ACD3D2EE 31C4714F 0B79F594
862B5C9A 11BD89F8 94ACD8DB BB310CFF 203CF0E5 41F0E2D6 138E086B EBE627B2
471098DA 388F71C1 46FD4B03 5EDCED94 7005BFEA 79EB30EB 4C3E82C F6877C83
```

t_5 =

```
4170D919 99B8BD2C 28CFC6C9 6682C0D4 570A2DE3 46734216 10F5F5E0 B8FE0C73
55B1A758 4162D5BF 3CA97EF1 C9E58D0F 59C2AC71 D293DC4B B79B80F2 53322535
8C93D00F 152004A7 9DA6B065 9096A5A7 12650702 E5871C2B 0A106CF8 7271B0D6
ACBCECCE 4463ADBC 1FD2A484 E91C4AB6 03F1507A 6B820C25 84CB3AE7 765194F2
2C1EDD8D E3A0BA35 449CD004 A5EC30EE F45230FB CECFAFA1 1FB8E637 FB9C2C75
C0B24140 C2A6CE20 BF9B569D 583C1C01 1F821EEA 16D3B792 FD33F388 E66BAC54
92FFA95D B2AC0CA2 5582FFBF 565C8F1C A9490107 59A6E101 9894BB57 C1A5A030
6CEE6326 EB35193E B245A3AE 829F6929 FB5EF611 0B84C4A6 03316161 3F998558
```

c' =

```
44F1BC8E 392071C2 AECCF97D 770750E9 15941091
```

E.2 Mechanism 2

Security parameters:

l_n = 2048
l_f = 104
l_e = 368
l'_e = 120
l_v = 2536
$l_{\square}$ = 80
l_H = 160
l_r = 80
l_s = 1024
l_T = 1632
l_ρ = 208

H and H_T are realized as HL function in Annex B.2 with SHA-256 as the underlying hash function.

Key generation process (setup step):

p' =

```
5E1C70F4 512B6FFB 527E1EEF B8AD51DA ED15ADE7 8D69DD26 5562F46D FEDA3D09
EBD6EC7B 7C6FB931 BE416E20 FFCB9873 D3DBF653 85897F39 729913C8 F453556A
D475CF65 3B861673 35F8E4D6 8CF9B2EB FDF1D9BE C5C0BAB4 BE487743 2946416A
CBCD4189 E78316FF 0B69E013 50216582 2E5CDFEA D7257FF8 D5F2E5EF AA5DAE65
```

p =

BC38E1E8 A256DFF6 A4FC3DDF 715AA3B5 DA2B5BCF 1AD3BA4C AAC5E8DB FDB47A13
D7ADD8F6 F8DF7263 7C82DC41 FF9730E7 A7B7ECA7 0B12FE72 E5322791 E8A6AAD5
A8EB9ECA 770C2CE6 6BF1C9AD 19F365D7 FBE3B37D 8B817569 7C90EE86 528C82D5
979A8313 CF062DFE 16D3C026 A042CB04 5CB9BFD5 AE4AFFF1 ABE5CBDF 54BB5CCB

q' =
85D9F41E EEBF0F95 CE53ABE8 BE7EDB94 FAF84AFF A04EEC40 8655B3CB 85856A87
20C662C7 B2C4D092 145C8D07 95A7685C 10B1C715 5F366DBF 83C5E4D9 5E311A7E
828692E1 3D5C04CD 0DF4CAFE F743F9EC 181DDB67 A4C6928A DADD9771 D0D5C381
4E557D5E C329E8D7 FC527B74 0A3A0BE2 2D6ADB27 FDE1A99F 87CE53B5 3841132D

q =
00000001 0BB3E83D DD7E1F2B 9CA757D1 7CFDB729 F5F095FF 409DD881 0CAB6797
0B0AD50E 418CC58F 6589A124 28B91A0F 2B4ED0B8 21638E2A BE6CDB7F 078BC9B2
BC6234FD 050D25C2 7AB8099A 1BE995FD EE87F3D8 303BB6CF 498D2515 B5BB2EE3
A1AB8702 9CAAFABD 8653D1AF F8A4F6E8 147417C4 5AD5B64F FBC3533F 0F9CA76A
7082265B

n =
C4D39A24 A01C9BD2 39EF0B5D A4F7E79D CECA6575 1DCF8EA4 4DECC07F F7F05194
C963B98E ACDC7C14 E6A808ED F656605C C73CBF69 22EDF311 9EA98591 32A384D8
E4A9182A FA926596 ACA4B3FC 7F77182A AF08A211 EE1751DC 2C2F8DFA FB4DFC5A
77193020 9E43D11A B8EB491E CCDE1F6A C1329B43 FB9F10E7 D6727739 10317A12
41DEB3B5 DE753A3F AC3C74FC 24715B04 33C27374 8F59155D 4600D232 A121530F
6CFC201C 4B6FCFD5 BE0E67E8 8B8B842E 680A07D3 2E9516D7 A60415BE 23D555F7
19048092 5E04352D 1B2CF4E5 6C8C2571 4AF8B6AF 09AA4DEF 1A9FC5A6 C1E2C144
FE12DF43 3695166C A1F7AE87 296F85AC A628DAD8 829B0CA0 333E77C2 DD761E29

g' =
64E6DC61 16767042 2DF48B32 B78491AB 12EA1734 D7EF7A79 86218A86 1A09E001
D933EFE8 82FCDF1D 484665EC 069C0C97 AE2692DD D78BD52A 6469149E 6A7B7331
7BD4FA92 6C08066B 7A0AAEC8 6A2A02CB 85C6BB58 BF149AA1 BAA17B41 33361362
3E8F2C94 22355488 A85314F0 65E7D0C1 00BE5F8B E548075E 79968B2A 72140F14
D106AE91 3D3DCC92 633AE566 F4B758E3 62989061 40B46045 DD0300BB 1BFC6D4E
F26C7C57 93232ADE 3528D8AA 10CD959B 72517C59 AE96D4D9 2F46BA29 FDCD5C7A
E6B54D86 11AAE7B3 1860E188 26C58B2B E3954022 F2462386 DA24082F 34F47860
D78F8B3E 66F3BB0A 21FB5671 9FC6D1DA 7BFDCAD6 8FFA311B 9F860A02 CFCF8EC3

x_0 =
2822EC39 810EA311 47D1B0F9 3E9C8C39 EA7D2BCD E23AC397 6CE86D02 D9953FA2
00E67F5A 45A82965 6037E9C2 B477AFC2 9CB15F00 A824F618 449613AD FD037152
44BD640E F8CAA652 C7761B48 68F57546 F21EFA78 4D7ED26B 665CCB10 97799AE0
BCB1EA58 9C33B8D6 EFFE4A18 5D20DE1B 497C528E 6ED1CEF4 890B0124 0DDF9171
179B0462 68700E4F 3BD8DD7A 2D714834 D31ED499 9B462578 3B4BBA42 534E9ED7
166EF766 3031BAA3 2F02EEE9 33F607D7 4649FE10 BCC83B84 B2EC2584 8F84A685
A70CE815 0A0A0CEE 292B5F80 C2FEF017 906B09EF 7634EBDA 7466BBA2 71A6AABE
01E0BD86 E6C297D2 3C355B8D E35BCB72 AAD6B478 F8461F46 DC8D4FC5 210ADA79

x_1 =
26107168 30EA074B FBB18879 847AAEAE 15D53982 09005C1F 9FD4D938 C63FC673
5CECBB1C 9C397715 F20D9822 A5FDF667 C1CD67BE 58CA72B7 CB1C99F7 09C4BD93
8754508D 20AC9938 051A57DD 56731135 ABACFAF7 4D12043F 474F0D85 FAB63E32
5EB44839 680F1F44 7C2BCA0F 91F5F7F4 929E5DA6 108A0FF7 F4044C5B 4D23B8DB
7F650442 73F3F047 36ECDF7E 04340568 F362E2E1 AF799ED0 5A79A89D 21B301EB
87957CFA 72C310FD 99108B8B 2F2468E1 215EB6DA B0ED8B72 2C8754D1 C06DCB32
37BF7D34 FBE15424 20476F1D 0D75867E 63DDAB8B 612911C4 5E174F5C 8F760D09
D2F70DDB 63948E0F 95BDCD95 2C452932 7D5C7C6D 30B4DB34 AA4294EE A7352255

x_z =
20812024 49E048FA D45C45C1 69CB2224 6F1A47F4 1331CF54 30CC8C95 36C69595
97AFC1DF B79C6963 EB61BF37 BC094F08 E593593D 43470517 18EB7669 442AC94E
DC1972C7 24355869 844A33AF F3A03F43 B32A5141 01627BCB 955EFE52 7418A02E
AC6A0D1C DF0D6C9F F672A6B1 65FA265B BDF412B2 DE025E1C C76E6A98 856B2D5F
23B8050C 9CCCF898 AB123B8A 05EB313A 921CF074 3A50B308 E9027403 A6CEA502
BFCEE2F0 780E7C97 F78BD718 6D378CAD 3308FFCA 16280EDF 8D87F640 D8ED1907
6CB35013 E5A0D0A4 5DFC4170 2A2E13AB 0C86B543 8BCFF6FB BE2D08D0 6FAB4FBD
AF4CE3A3 6BA4ECF1 845D7DD7 6602C294 6AADE482 1DFC6902 A041DDA7 DB2306D2

x_s =
2DAB31AB B537B268 4C687EAB 9CCC355B 60B66F3E 253EA955 81C9043D 843DDF13
202A43C1 BCDC8471 850C34FB 18B79C43 6C114B61 D58FCBAA 09F0E8E7 A8FAF52F

```
31E3CCE3 BFEA1B1B 66758C52 9F50E212 57FDB83E 026E202A 9F16103D EC40AEFD
77FB6FCF 669050EB A502FBA0 BD80DA77 DBE758B5 800DEC93 6DEA6A70 10617392
1AA1D917 BD6013C6 C5BE671A 6CAB0138 835DF214 B09D8665 C1F30292 7E6DD6BE
CE71E44F 7048899A 2C822ECB C21E5E11 03379505 81AB1E4F 43E3921F F8ECEC9B
D1EC05EB 83701D58 464C47FA A6927914 B00BCA26 2F76A740 070F6CAE 7849EFBD
72BFCEE5 18B8C7F0 3FEA4C28 06224159 52234AFD 36100C99 5B24F95A 020002AB
```

x_h =

```
1484FEE1 A6F872B6 367D9C8D DDC47203 1B84B266 EADAD7C1 02C901AF 1B8AB266
348202E8 A7D252BC 5350F078 502792BF 40E8F640 A51E0F95 E116AEDD C3CDF304
6D846297 176D69BA 0A529214 B111A204 E399C908 AFC94D47 D12FC8EE FD80A7CE
BB0F715B 943F844E 2A0E7647 A82C20C5 DC0444F9 F9413CB8 71DC0DE6 9931FD70
E4E9A917 47515A74 8BFDCF1E F965F9D4 F1F51B20 B8113012 FF0B4FB9 197483E2
79015452 D716C9B9 47853A65 8D0B2120 E626BDF0 250BDFC8 01A8DF42 32DE93E0
4C055A40 21DC0ED1 18F6879C 5F146580 2C2919BB 6DE5B66D 8C59DE3A F9E88944
6C3435D6 E6C2CC59 69EFE106 BB74B831 7DEA9EF1 301B264C 30731B74 73718637
```

x_g =

```
0C8CEFB9 5A4D8899 EA18A825 7D1657D2 44C31B6B E324116D CE418482 D8DC479C
D88B08EC E085C363 67BFC320 92277C2B 740D3D2E 94F4F946 ADEE423F 319FCC1C
34216692 1BB72869 2C71E0B1 59A5300C 59FFA16A 954F2F3C 54BD58DC C62885F4
5CB24392 D2F60CD2 3E597D2E 679D4D4F 3402ADF4 A59717CD E56118D5 2603C99E
3CE610A2 F07C687C FC546304 5C6D1103 53541407 B2ECC1A9 B6DDE98E 3CA27D62
25A22CCA 15533AF1 DC6B0AAF 555612A9 7D51C784 D882306A A8029474 C58E2E85
944656EF 9CCEA62A D5514259 1ED678D6 2FEB41DE FF2ABA0B E28D8AE2 7B51C16F
5D9E997D 69C6120B A07BC137 D8340A3E 2A99CDDA D4E81E47 38E77C3B 8F07E2FF
```

g =

```
B37D2E55 1F0FB3BF D1D132A6 DE1FED15 4D1B718C 30B0E6A4 A3CAE0FD DB10D499
F5D63330 59BA3B8A 94E77BD7 6D35AEDF 4EF9042F CBD54450 6BEBA44B 7447BD6B
EB11C6C9 2194DC11 BDBA2791 7E540972 2AB897D3 2AE3A131 5FE38E96 E7BBDC20
6A25904F 613C4FC4 D73BF059 0CF467E2 020FA2FC F84C7F93 F93C91B9 C9BDDE03
672F0DCA A747BA7D 6608F9EC 3D02873B DB0F1465 1825EACC 03E822E9 722A00CE
807532C2 D3941542 514F4F36 193CC7FB 4211D439 7CF8D9BE ECFEBD82 37BA2416
F85652EE 88C3487B 90C27443 9220907E D68E66DF 4232994B 0CBC0AF3 B721B21C
20E001C5 BEB470C6 E5EADFAE 52718308 0C190D20 4E75404C A280B591 B91541AB
```

h =

```
6E4DE4EB D9B9DBE6 E6869FDC 48123824 445D2BE7 3CA64B9E 085C1DA8 CA19C956
75B1AD8C E877A95C 4ABD4ADC 7E5687C9 051BE278 CA95CA02 59D7ED9B B4E2161A
22973A3B 6284B342 64217050 61995D0B B35A9AA6 FD85FFBF 7EEDAB5A E455489F
2B7D2679 8936953D 63AD4A8F E60D0AEF 07761E4D F54CD61E 4C4948FD 7277C7F2
8CCB9E89 D867245E 684500B8 7CDA3B56 BC381904 A505BBF9 FFE79ECE 308CEB02
7D3386D0 24CEEA34 54D1BD25 4D5E5FE7 23840F7A A6F38E34 943177D4 EF44D807
342E989A 622AD67B 03F92BD7 B2A888FA BA5A18DB FC0712FF 89DD9749 DF19B55C
CC88D01A 037D230E 2A0BEDF5 E53AAA82 E4A3985B 0A3A76AF 2B71A1A3 E7A32828
```

S =

```
740A7C2E 69C11000 5E984651 52D0C3FC B7CE5F02 46273EC5 0C75285C 8B258170
72642A07 ECA4A07D CBB1F401 A15B0476 B2667F4C 40202BEB 90655CF9 7E3797FD
D4E81D82 EB663575 B91E6328 7CB66796 1C1F649B F1D4DF5B 86872A70 0C6B3623
76B29232 4A7E2840 3059E2DD 18D4ED7B 9DBA49FE 10AFD6C5 9184DC9E 5735BAA6
90581346 6F1C8319 FE6C239E E0D6373A 5029582A AC973C9F B264E7F4 596A7898
709575C6 8E98F19B C9100DA1 5D5D8B67 C63D7A2B 141E086E 21385AA3 FADE7EA9
F7BEA90B 87D62872 E7C0AF10 EF3BB3E2 3C717CDF 41BD2C0D A59CDE40 99C8B49E
FB4C5B70 1BE3310C 7B248706 DF33D1B2 4596CFBF 86132157 8D05B811 8C5496EC
```

Z =

```
1DC73C13 2C7B8D0D B4B05912 3A0B9007 4371BA5B AE93E1F9 A6D03A6B E7382F52
9C01FD94 376D4146 64FC768E 30980362 0D990343 84A5EC50 29A874BB 34BC840F
A5838DB7 458A4158 CF764E54 997A188C 8CFC9BB8 BEE2D7EC 813D0D9C 80996BA7
83DB090C 79A3EEE9 E811CDC5 31450218 210C3C0E B4E8A9B9 0B0BA5F5 1F42114E
9799302B 9CB423AB 6C223BF2 4C7F6CD4 3020A609 148EBB9D 6BC51CA7 24404F52
79F5AD7C 6C6A377B 3E14ED0D 67CA0C7E 7B72447E CE011200 52A32706 B41D6B9E
ED984342 5B60AA4B D8774441 DAFF94BA A1906C34 0E9295F7 280B5EA6 EFC0E1E8
20D9F90D E5F08A8F A7AD42DC 33998790 F4D7BBAC 5E766751 FE3C21B6 A7D5E6EE
```

R_0 =

```
36F47840 BF894A72 09DEC142 3A8A8A2F 6942EE22 DC6F8163 C464201A AEF98E0E
C5009F6B 380396C2 A68A8B0B 8E057E84 187787E1 99A0D69D 6A66E8A6 1C8C80BE
49A2002B F3E7D24A 9890DF46 ECE9C632 36132742 3FE46ABA 5BE5DD9B 583DC624
```

E6C9480B A123AD5C 717EC19D 35378DEE E8CCBC56 840DA117 5129FAF2 4A0858D9
974DDCCF F160D9B1 9A3D006A 3609DCCA E0EB9BF4 728F4D69 7A68B282 D6E24379
D542404B C76D41AF 7A4E7206 2F43CE03 BF66870B 3B135176 19A048C4 A4D55EF9
DC2C0981 EC27155D 29494732 417E5EC5 4AB8E721 286DD632 1F765E99 9115F52E
705BC800 8A308168 191AFE52 30554A97 9893D9BD 8D960E25 4717CB00 4B0C0032

R_1 =
A2F6EE15 D0267740 1D2FAF78 09F329E8 4DBA4DE0 3837B8CF 6C63BB0A 4FC009A0
44000785 32F82E43 02487BE0 765B297D FF344E46 99B6E993 0019BB02 5992212E
7A3E68FC FD2751D6 5AADE620 A31D0FC5 52BF579B 354F29C4 5F0EA2E9 CD08788F
6FA10EC2 16206626 3B3FF615 58915213 E259D259 1FF9890C 5CB2E2DD 0306ECD6
B8DC5C6B D8F2E632 285DCC71 7DC49021 0ABE9971 828A419A B27BF5B3 A7004FA5
D6FF78F5 C3992A5E 19A98ED1 6553497C B3AA98C3 A04F887B AB1B78E4 7DE31EFC
D8340D62 7EBD1F8F D22FEA81 460B22D7 86C712FC A5313F55 B355336E 83594572
C0F2210F B5D31680 9C5D3BD7 AD71AF03 B6E0C3E3 FD3AE398 C5118F15 B5E2D875

ρ =
0000C510 A08B8603 4D27F9F5 6EAE0EAA 644208EB F7BDE940 2BD8F59F

Γ =
D857B2D1 92A8EA44 37E8BC76 AA9F3BA8 B59A9E13 F77BF943 A32792E3 5E7CFFF0
EAABF964 742CF942 882DF791 238C464A A3C09794 B411AA9E E9DF212D 6455FFB9
BE820F54 54A15324 EB685348 C09ABF73 CF6AE657 361DAAF0 95A3D1D9 20DA245F
4A86B55E 405BA46F 4FFC2681 96EA5D37 4CA033AF EB40C21D 2582F875 103776B0
130BD921 8EA679E2 E75A8990 1449E939 907BF63C F9CE715B 87E4497B 44B3E21B
A8F4461C 194C350E D9EDEF8F 243569E3 2A3567F4 6EF0EE7C B75F12D6 759E413F
120FFE75 B70AE8FB B59BA2A1

γ =
8A44F4EF B1679D6A DCE098D2 34996CF8 52D8732F 39BE767D E4A74D7A 7CD2BCE9
A63B5332 5F31B3D8 047376BE 230965F0 31B37355 8EAEA0A1 C18AB66F 04A7D3BD
3E4DB48A 7F931526 184BFE97 6B621C74 1590BBF5 8FC99E27 06B5B164 E25F847A
6D80AC4D 144C47FB 26B45F63 37CBC354 73F6D27F 8FB7B88D 796ACF20 8CF49CEC
6BCC0DEE 5DDFAFA3 77835054 84DFC5D7 2474D13F 6D3BF05B 3A7B141E FCE8E665
4723080B 61ED2022 8071E7D8 ED52A149 8EA856D5 556DA256 32728402 663AE107
169C0556 AF2B4D9E 3A3AA0A9

Key generation process (group membership issuing process):

bsn_I = NULL

J_I =
A7F56F51 D1C0F7D2 256FDA75 8005E87E ECE23080 276F9B20 96065A47 404486C3
3921EC11 0063BADD 653F9D14 35C742E7 50FD4034 5DEE9C18 720C3775 02F4E1CD
0765160E 5BE01153 9E356EC8 A06CC9D5 D53E1985 EB545DD6 24D62308 D6555B55
F5EC8545 61DBFC8E 042BD9B0 709BCE98 022F2FCA 45627904 04390C09 96B0095A
714609AD B51FC494 97BC3FD1 2BC42CF9 27C1EB04 90CEF1BA 3E6E9EF6 6D074FD0
0AE3A2AB A30F974C 26A8D287 12DC0015 835F600E 6975450D D3563EEC 0380A955
53B0C8B1 C735FD92 A3B5650D

f =
00008164 1DE728E6 40EDF561 B2E37604 6D15A72C 3814E51A 7D380AF2

f_1 =
00000081 641DE728 E640EDF5 61B2E376

f_0 =
00000004 6D15A72C 3814E51A 7D380AF2

v' =
0000FF15 694264CF 86AB92F8 89D80D24 1234E15D CAF0797F 0A4CA49B AFDF91E2
3C102A43 887FD7EE 96DC3299 16CACC10 7FE16318 409829B9 0CFA0C55 7F5B5B7F
48CED6C7 8461FC69 7B52E170 0A916A0C B41B9E3F 102CAB4D 334F6975 90E905F9
31A7C841 5BE5F59D 44F29C28 7EFC9BD2 1690014B E9D0928B 12C77AB4 58F8429E
A82420B5 859397BE B719432A FB2E7B37 B286DB8F D3A15167 1CC26834 C04F3CF0
CE42A1CC 9ADA165D FD80A7F0 0DB1077B D45B5465 35C3DFB3 78C61CFB 5A5932C2
F56DBF0D A27D7C44 05C1CB0D 05108565 4D33E299 52B176D9 6DD86798 A9D1C5E2
DF266D49 60F01BF5 1F47113D B8925A30 716F5BC4 6D0FEFEB 299749AA 7378372F
C167F8A5 7785CED0 BEB86F99

U =
BBE3AE81 4D301822 93CFDBCE 8E93F383 7156AE1A 1AA38C6A FB3A905E E8E662EA
366052DB F759DD03 C52FE609 8D69B666 79C26AE5 BF7E420D E86CF189 2E928751
E17F1232 0BE0F221 A2E5A4CA 45D4234C CB5E4A8D FEF73A3A FF9F910F C346915E
F2675DA2 FE1DC06F 0A6915FD 552508EC 6728A0A2 B31BFCB4 97319ACC 1F9C8F59
DC1175D8 5EE148B1 543417F4 E59F2269 894A9DFC 4F7433B9 80B97591 2B8F714D
463C08F1 76C9B387 FC7D1C38 E5BA70FC 3EC5BE1D 50F6367E 677AA3C0 BB15BE71
E11354E5 3CFE0200 06097D6F 10862577 6D4A5E8E E4E0FD70 D86970E9 0512BDF3
4F828A30 238AF901 6BA60A50 DF230A9B EC645504 D0189730 7F153AD4 C3B1F027

K_I =
3704419B 9E4B8439 B804006E D5CF3740 F49D6C87 D7BB6DAA 3074890E FEB287B5
43B484AE 45C3939F 07547B3E C8AD159C CC2D1361 E4523550 C06630A2 EE4CFC2E
D84989AE A52209A0 D716648F 8ACEC20E E9AAAFD2 A7D2F992 EF9434E8 8837CCA1
18957AF1 A77388B6 9EB7FBDC C8A10DBD 213F19F6 190239BD 61FC46F7 FEE8829F
D7003017 895F9CB6 18DF288E 6F67F5B7 D690DC90 B3432F27 032949D6 B88E8A63
2491EEE3 F38DF476 D245C62B B901474E 4FDD53F0 A3F9A454 3D1177D4 1493844D
AAF14EA3 91697E65 A4C83670

r_0 =
007890DF 082FD00C DB16176F 6E7CDE5F D085B4B7 0B2A32BE C2ED7379 15288786
8DE01D77 6921E34F 65AE51DB

r_1 =
000A3C90 1A49433B F354E9EE 7ED5A3CD 662441F6 58D0E074 17F930E5 4B332125
82548ABF 532F7074 D47EE311

$r_{v'}$ =
F82916F0 B76C005B ECCA530C AAEEAB6E 6CC81B49 4707EC20 A21A99B4 9AB4EF88
FF56F737 532A5977 B9D991A0 7AEA32C0 36EDC5E1 ED1DB86F 2BFDA7E7 EB4EE4C7
17E06754 5951FEF9 43E86831 04199E1C FC59AC53 50F45B78 AACCB608 4326ED2D
F525DAFE E005E8BE 7A4EA44B B3A93DE2 ED6DD8DA C083019B 9E6C03C0 F860EDCB
6B7386C8 649AEABD 7E1F7980 3B915405 0DE75DD0 88C27677 0CFFC958 CC1F1396
B223DFC6 22E4DE52 D1854E28 00748173 41C0E9F8 2438F8E4 92B2BCE6 6ADFE6D3
D4D46B02 602FC0D7 A496F107 91F80B08 CF25FEE6 89B312A3 4DB72AA5 4840CD8E
C7FACE0C 51E62D7D 7FA16E08 C2A596E9 D38333B4 753EA849 996F0EC1 B926C69C
1164F27B 14782025 5F6BBFD7 8EB3AB87 92378C7F DEA60A7C 70968961 EAB77B3D
2BDD08E1 997FD092

U' =
9DB5BF27 AF71EC30 63166E9C 83038830 915FDC04 2079E405 4C199118 BE1301A2
EFA678F1 6C173D53 D6A32A7F C8B349F0 CAEB0DA2 146451B8 9A906844 89470741
6DD3656A 15A96C4B 43C01DB4 CA56B994 6658BF5F 7A4F2DB3 0B9D87F6 620AC8A2
7677C4E3 1306C164 3B65B471 D0361AD0 2EB2862D 1D8AFABB 1915A987 A2C78102
3E8BDB65 0F26FB2C 5B34C149 731859E3 79E8B552 C56B5835 034ED0E7 E1F79EA5
D67B6689 37F1C8C8 3824BC07 0F3D5246 84867A00 E520AFEE 72BA73AD 90B3F81F
32284D81 96E73C66 2719B8DA 949FFC8F BD45E98A 1A3D034D B38E2059 B5D193B5
9278EF21 53A64B4E B895C8CD 9A6BB9E0 1E1421BF E116E67E 22E291E0 7EA4FBD9

r_f =
0A3C901A 49433BF3 54E9EE7E D61C5E45 2C71C665 ABF68B87 67ADD3AB 03A6DA39
5FB4F211 F25DE84D 94069886 8DE01D77 6921E34F 65AE51DB

K_I' =
CDDC3235 1F12CEE9 59A8B63C E23571D5 5136F69A 839EED2E 510B4F05 B1B3CEA1
6B92FA5C D8027152 4BB1BF3E 6F8C0EB9 B6E9B085 F4745615 5DB85444 72067499
AFA2CA09 C69FFFE1 2EDBE239 D52C6E0E F5996842 0D6DCD1E F6F6A180 A1D77C02
180440F6 D0465CAB CFF7D1EE 073C4634 83367F5A 89D2E067 B66757DF DA493EEF
87810E40 D0047321 4E9ED986 926C182E 6545BBA8 97BDE0FB FB60E6FE CAAB3622
7E1FBA21 3BD2A75B A2505A4B 0745DC5E A55ECD61 A871BF3D 42AB7E9A 0A16B42B
CE017E82 94BD5E73 3D2FF022

n_I =
D9019EA2 1EE2A92C 34AAB0B2 8A1E77EF C6C9280A

c_h =
5A9DD72B 612E1B5C 5EEECA08 F9E82C93 1F889A46

n_T =
00005C09 900EC474 B362BC52

c =
15F2C5DD 09B2526E C45B06A7 60AB4976 9A1F2AD4

s_V' =
F82916F0 B76C005B ECCA68EB 53F6579C 249FF698 DE2E1363 AB4DCD9E 8A2C72AC
0801371A 36818E15 828C3439 6FE77A45 7F273EB9 367F5EFE 7C6E1935 A096D869
7B94AF10 42C97D9A 32ABC44C 958BD514 6F81061B 163A2C25 47F42A3C C1C417D2
9341667B BBAAFA86 DD1497BB 5971AEE1 8FCD9E60 A77A446C 6EC31842 4A71142C
BC415F4A 3C6636D0 983CD57F B1211821 609F1D58 82D57A2E E2585944 2DE2E1BF
7A5A028D 01E2711D 3B986C1A 56C45A89 4F17E138 46856ECE 9CF71B1D 4ADC0E98
7442A3C8 F89170DC 3939C3CE A57C6E05 75AA14C0 D1FA3BDB 372D6C5E A64D6E76
DF6E5182 6CB67274 2C3CC5E0 3BCEF5C2 E452C7C7 D5239132 72400E8E 7B0F04BA
88D7AC45 B169B45B F7036591 6EE111F1 AC26595F 84FF75E0 933C9E81 02794DF3
24A5AA07 5A125546

s_0 =
007890DF 082FD00C DB16176F CFA22F52 792DD061 477A8688 D33F5E89 1EA82E05
3BC6CBF5 B828E3B5 31311643

s_1 =
000A3C90 1A49433B F354E9F9 96C0BDCB 9D0A9A54 2B0BD0A0 F807F56F 61467552
A8C8C522 509EB193 9D3E9CC9

U' =
9DB5BF27 AF71EC30 63166E9C 83038830 915FDC04 2079E405 4C199118 BE1301A2
EFA678F1 6C173D53 D6A32A7F C88349F0 CAEB0DA2 146451B8 9A906844 89470741
6DD3656A 15A96C4B 43C01DB4 CA56B994 6658BF5F 7A4F2DB3 0B9D87F6 620AC8A2
7677C4E3 1306C164 3B65B471 D0361AD0 2EB2862D 1D8AFABB 1915A987 A2C78102
3E8BDB65 0F26FB2C 5B34C149 731859E3 79E8B552 C56B5835 034ED0B7 E1F79EA5
D67B6689 37F1C8C8 3824BC07 0F3D5246 84867A00 E520AFEE 72BA73AD 90B3F81F
32284D81 96E73C66 2719B8DA 949FFC8F BD45E98A 1A3D034D B38E2059 B5D193B5
9278EF21 53A64B4E B895C8CD 9A6BB9E0 1E1421BF E116E67E 22E291E0 7EA4FBD9

t =
0A3C901A 49433BF3 54E9F996 C1365C7C 12CA2437 E6E6B867 D7979EB3 BFA3230A
103FA8D9 71F0F226 5D44F705 3BC6CBF5 B828E3B5 31311643

K_I' =
CDDC3235 1F12CEE9 59A8B63C E23571D5 5136F09A 839EED2E 510B4F05 B1B3CEA1
6B92FA5C D8027152 4BB1BF3E 6F8C0EB9 B6E9B085 F4745615 5DB85444 72067499
AFA2CA09 C69FFFE1 2EDBE239 D52C6E0E F5996842 0D6DCD1E F6F6A180 A1D77C02
180440F6 D0465CAB CFF7D1EE 073C4634 83367F5A 89D2E067 B66757DF DA493EEF
87810E40 D0047321 4E9ED986 926C1C2F 6545BBA8 97BDE0FB FB60E6FE CAAB3622
7E1FBA21 3BD2A75B A2505A4B 0745DC3E A55ECD61 A871BF3D 42AB7E9A 0A16B42B
CE017E82 94BD5E73 3D2FF022

v^* =
0000002C 84E2EC62 C248199D 9F443C07 7BDBAAC8 86BD4C4E 92A01202 6C0134E8
8EA8D5E9 B09BE372 925A3A49 1608F4C0 C17AD3EA B200DC2F 7690D690 1C27AD6F
95B3483F 3904D382 9CEFFC8E F7F75808 A0D66B02 B0F514B4 139887E7 6C195862
C2FBF920 E6E29A2E A0DE2A4B ADF53A33 1EA61F76 AAB76BA0 C302CA57 F2877A8D
8DBC6EDD 47D68D3D 3AA0E7E7 3DBF3015 CDB255C4 5C135C09 0F8F27B3 BA640486
92BAEA76 780430E8 223697D6 A4997D69 E6D2B2EF E70503B7 665EED87 BC7DC7AD
FE4BD40C CE0E5B72 AD45C8B6 553320B3 4985A722 2A5F2045 813407C0 0FC81E1E
8F856BF3 988EC80F 32390BD5 E284BC8F 4394B9FD F886CA6A D04D62E6 D087BA7F
8F686379 68DC7C64 DFEAD7B3 2B25F363 2AAACD91 B48B5D39 15D74EC0 EB9D6B3E
01E8A36F 9CCE6C54 7AF11121 8339C474 AD769D53 EC62C572 57C39DB5 3AD23B30
e =
00008000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000B7652 6555893E EFC8FC96 F6171EC3

v'' =
000000AC 84E2EC62 C248199D 9F443C07 7BDBAAC8 86BD4C4E 92A01202 6C0134E8
8EA8D5E9 B09BE372 925A3A49 1608F4C0 C17AD3EA B200DC2F 7690D690 1C27AD6F
95B3483F 3904D382 9CEFFC8E F7F75808 A0D66B02 B0F514B4 139887E7 6C195862
C2FBF920 E6E29A2E A0DE2A4B ADF53A33 1EA61F76 AAB76BA0 C302CA57 F2877A8D
8DBC6EDD 47D68D3D 3AA0E7E7 3DBF3015 CDB255C4 5C135C09 0F8F27B3 BA640486
92BAEA76 780430E8 223697D6 A4997D69 E6D2B2EF E70503B7 665EED87 BC7DC7AD
FE4BD40C CE0E5B72 AD45C8B6 553320B3 4985A722 2A5F2045 813407C0 0FC81E1E
8F856BF3 988EC80F 32390BD5 E284BC8F 4394B9FD F886CA6A D04D62E6 D087BA7F
8F686379 68DC7C64 DFEAD7B3 2B25F363 2AAACD91 B48B5D39 15D74EC0 EB9D6B3E

```
01E8A36F 9CCE6C54 7AF11121 8339C474 AD769D53 EC62C572 57C39DB5 3AD23B30

A =
B7F5B65D 5D71E0C0 C8A60A74 4C63453F F4B5423D 352F5994 A74B4121 20868B82
E87ABE07 C1EA3395 1A1FF0D6 9EF7B9ED 55F232E9 CB47E572 2C852D17 9A7DEB66
8699601B 6D4D1DDD 31D6B8F7 7F6ABF95 C4D7715A 78DA9A87 D9F7927B 0DAFCEBB
EF1B431A 1E2FC235 FB6A9240 19EF4D19 A7217279 8FEC6CC8 B0E2ECCB D77C0F50
A7A3404C 5D6EB115 0ACF6BEA 6971DD6D 17DF3C16 40E8832E A3E067E5 BDC13BED
6BBACE61 FFC03E4B 91B43F16 0D242E1A 857BE481 8DE063BF 90475B53 0418272E
AF2043C0 7A5B7145 C172C2B0 D14BE91F 11BD808F 2768EC4E CF0D26D3 DF532AAE
5A33CE9F F1354085 10D23BCA E2D64673 ABBE7B4B 6F7FFF9F 2B0F29F7 97AF713B

n_H =
00000F99 66F0E10A 64460CBB

r_e =
09C9F6CF 8CBFA6CC FA0E2E72 DBB990AF 4839979B 52DE73D2 D44F88C2 08871E12
7703FE19 992BA1B6 76E38940 7BA0B111 106AD846 D913938B 40080BEF 6E1FFA36
83E0C5DA 0F71C9BB A5EC85AA 1760EB42 7B7F1674 CD884011 16F9697E 502C6D5D
452D1B90 76437A6B 78272DED F8AFB330 E2676822 4033108E 7BEACA5F BFA41027
C72FEDBD B9F14096 18C6F5C4 BF84226F 79F675E9 EC886615 6E917414 342813AC
C74A69B6 1FF0BC48 17F53769 00EEDFD9 0A818FF7 93DF2A5B 657B0C84 6F82D80C
FF8DF797 3DC24969 A7C23B42 432AF901 FF988851 929BDCEC 98B3DFC9 44DD7071
7AAD7BC5 323E2FC7 54234CB5 BF292BFA 8A03F437 D5ADA7DC 42E5DBEF A4A45453

A' =
936B6F40 16C9D4BB 006A372B 3A79871B B87B29B8 F43F9335 67CFE0BB A27095AE
2C415419 06C315A3 C0A966C5 90847A4E 6B7C0FEF 5C9FDE67 9441D9BA DD2B9863
9F267DEF A70A646A 5B20E50C 34308A72 0508CA75 24FF4176 DCC865B3 3DF0679B
1C5F561A 1251FB4E CAA98D09 B2D7C977 1A13B097 EB713823 9A58A985 14B80248
5482E549 4595C4D4 1B1B6E5E 270FC8F2 3B276915 8C404078 68F69173 F41D2076
48130EF0 3F9478CB 8AF2C8DC B63661C2 B3519A11 F55C124A 121C9A36 D138AD2B
5E9855BB 02945B33 7BA8AFD9 553B415A 68191335 FEE28B3A 6A39A8CF 782EFD79
87241452 B8DF02CF 9A2D6A20 4F5FEB15 6E6D5583 A4EE187B 266C064C C19D165E

c' =
948F94E1 6ECBC608 091FEDAE C93C27A8 090EC402

s_e =
276486A6 7A99DEDF 1E7B32B9 7F08967F 73C6635F 7426F178 2D721242 D1C3E71C
44D834CF C1BB56D4 5677DCFE D0C590B2 95DE1EA6 1B785F76 80E66819 F1146B72
BE6AAF67 AB54620B A8F0ACB8 92E644BB B5F23149 525C4841 F3119F45 9CF1969A
5DBFB80C A95B7A0C 2C1545AE 26E2A1CC BE417A37 EC905729 69A98DEB F4B10D7F
1E016897 FA2ABCC5 8DF5139A 01E59363 F85D944C A6DDE51B 63E7DBB5 76D1A19C
B301A3A8 5E3F1489 F404DC31 CC551F43 B9DE2701 AA70A4F8 A2B02599 4BDFAF31
C25120F8 68E81114 A8781D68 DC2F7621 17A86854 B284FDE3 908AAB2D C7186230
49DB87DA 0A3A25BF 12E1469C 9B8A1A5D 42B5C494 8C186CBB 84A09071 9BA1DFE6

A' =
936B6F40 16C9D4BB 006A372B 3A79871B B87B29B8 F43F9335 67CFE0BB A27095AE
2C415419 06C315A3 C0A966C5 90847A4E 6B7C0FEF 5C9FDE67 9441D9BA DD2B9863
9F267DEF A70A646A 5B20E50C 34308A72 0508CA75 24FF4176 DCC865B3 3DF0679B
1C5F561A 1251FB4E CAA98D09 B2D7C977 1A13B097 EB713823 9A58A985 14B80248
5482E549 4595C4D4 1B1B6E5E 270FC8F2 3B276915 8C404078 68F69173 F41D2076
48130EF0 3F9478CB 8AF2C8DC B63661C2 B3519A11 F55C124A 121C9A36 D138AD2B
5E9855BB 02945B33 7BA8AFD9 553B415A 68191335 FEE28B3A 6A39A8CF 782EFD79
87241452 B8DF02CF 9A2D6A20 4F5FEB15 6E6D5583 A4EE187B 266C064C C19D165E

v =
000000AC 84E2EC62 C248199D 9F443C07 7BDBAAC8 86BD4C4E 92A01202 6C0134E8
8EA8D5E9 B09BE372 925A3A49 1608F4C0 C17AD3EA B201DB44 DFD33B5F A2D34068
1F8B5563 4B39B4E0 67E0760E 0243FCA4 50B5FCE4 ED053EF7 9C185FD6 02F58AFB
D9C6C531 66C3FD46 E1765404 BAEF4688 9E017AF5 F3864268 4764C6C1 6DDA5BFD
984DD8E9 FBF22B7C 4ACD9334 710E998B 5E9B5BBD 8DBB244A 6B751D50 FF56A0AF
11B78648 8E943234 0C072A61 B760F81E 3FCAF58E 8F29246C EBF28546 73970AD8
F97A4F44 80953702 80E71A1D 71F588E8 09D4E412 F8A1C212 1C0E1E1E 0D48C60E
9D36736F 6CEA1C74 67FCEB89 5B4AD98A 9DEDECC0 EDF48978 72CADF2A D649858C
9478E8DE B6105EFE 329C4E8C 98FE5AFB D47C9374 93B1CA82 76C76AB6 0AE47C7B
BA7AFDA0 0E3DC818 E801010C ACD10E1F 20EED483 ADCABE17 CF496C85 F98AAAC9
```

Signature process:

bsn = NULL

n_V =
05737FBA 8F76E72A FC558A5F 0CDE760E 7ED798A8

Message m =
"abcdefghbcdefghicdefghijdefghijkefghijklfghijklmghijklmnhijklmnoijklmnopjklmnopqklmnopqr-
lmnopqrsmnopqrstnopqrstu"

t =
0000967A 2B625B5F 5F3129CE ABF775EA 2E266109 CD2CF2A6 2FA2CFE1

J =
945CA21B EABC8BCD C87A1E6E 52AC249B 829927C0 FAE67253 33AC204C 5F11983D
D128CF3D C6A357A5 550464F3 01F1D063 0A2A8A1C 11D27684 0472FF17 4EBD117C
378B9591 13723D93 F04B7D8D 7E128168 93BADA28 1D598B82 6F12227F 29C0D27A
399FDF67 8F240EC1 20D7C53C 9A02DF13 0BD5BA70 51AE30F0 1CD2100A B69D973B
61A674C5 A2B84D36 C9D054E0 7BE46078 B2317BFC 97CDEC09 E4CD4D8F 22EB8912
211D6946 031950D4 ECE23F5F 03A487AC C68C5B40 C8112524 6D8624C5 AEB16C75
5478E2E2 7F215FAD 8BFC48D4

f =
00008164 1DE728E6 40EDF561 B2E37604 6D15A72C 3814E51A 7D380AF2

K =
77B21B51 859D2D33 19F809CF 20207D5E 3B068C6D 4CE8DC1A 93538179 F509ABD6
0A4C067A 9F30C3BD 22F5338C CADC6BCC 4CF3BD97 6795B412 DD500B6A AD0299C6
95C939DA 3A0D10C2 A08646C5 66D941DD 7C8A2E38 D379D25F DB68BCA1 FD6CD239
6EFB897C E528D00B 9F4C11A3 181F6141 38E75934 1C1F9960 3461F287 7AF0BF7E
CA611F21 4B62ABB5 B99C4B9F E803BC67 49C1DAAD 5745C009 B85FACF8 C98376A3
1FC95709 DC21C8EC 0181F46D 4573386B AC02A5C4 D3A0EB0E C44D525B 6406600E
5F30C50C A6DB57F3 E6FB9390

r_V =
0030224A CA1F064B 494CAE6C 0070D07C AE8DC01D A6F61A8E E97E4060 6FCFCACA
8A46C8CC 898B444F 73E1A0B5 5F37580F 78B687CF 1D76E796 7449AB2A 3B3B3A39
13457AFA 94623A2A A9A8D25B 657051C7 633CD92C AE4C22B3 FFC2406E E7051B8B
649959E0 D4EB0A62 4A4C7E6D 5CF87791 2F735F76 A67030E9 E1724BCC A80EC683
059E3574 D7C1BC66 86F63DCD 4731CB1E 12CD38B3 DC634C36 8A35F6D6 DC64AD5D
93EC3BF3 B9CBB16C E11148ED C73D8CB3 0030385F3 016223A6 392145F5 A8745796
0489DCC9 EBE6CE7A 6340515A 56CD3731 14B0EA5F B9C113A0 6D46C2E6 9BAD5CFA
442E502E 7A5C1E2C 301A4208 5925517F 29CB9E3E A97701AF 64B7EA32 2F6237A0
796EA8BC CBACD0A6 5C9B7989 4A696A11 67C09E2E DD0AA8F7 5AECFA81 269AB61B
CFCB6A15 40685DAF DB5E6E19 AE167723 6EC29905 EF592CA0 FC710D65 6CEF3557
E5633531 72C77867 D4D815BC DC48CE08 D96B6E9D D31DAC62 755992BC

r_0 =
00389F3F A1DCB5FE 13CB0C60 0D69D83E E07071F2 C1EA634F E9EC6401 35AF4570
F2CC73BD B5CCE8FC 99BFC702

r_1 =
00E58F41 A6E84CC4 C83157D0 11A2DA28 1CABD589 7FB752F0 B6D77315 7ADD87AC
BB1F8C26 77D14D7B 54A8125C

T_t' =
3163B89F A928686B 41FEB879 91128972 257E6226 E14A8A2A C674BCEA 7F16A14F
5744E4CE 7469F754 1C9F103E A21F808B F4EA958D 5AC74C90 94A3E106 6FBBB863
344A5C67 AE646870 736465BA E5C27D0E F142136E 618843DF D28D3F62 F6CA7AB6
0032CF6D C03040F5 31B6CE4B 63544838 840BAD38 FC73D30D FDA3178B 64433BCB
1318C9D5 E377C1EE D9386821 1673A983 E377417F BABBF32E D07DBB35 082869CA
B11921D1 B0A65913 8A1A9083 0064AEC4 08D8F2BF 8905BE7D EEACB666 F852A060
E46ACF33 4167967B 26C6FB3E 95F2DEFA D0B003A6 A483B46D 246F175B 1DAD1509
5F8C5B33 697DA579 6E722F71 DA7EAD57 147F3029 6FD9D5E0 F494CAB7 1EF939D7

r_f' =
00006E62 320143D8 BDC28F8B 08E8B2C5 C84D3CDE 303BE097 2E13A494

K' =
A4F9D8A0 01BA8A1F F0BC8652 4F446D25 6FE30666 C77ED82B 0D380009 DACA9DD1

```
45BF5DE7 D3F1E2DF 23E02917 2DEBA8CB 0066415E 0B3BDE13 D6936459 58D9A421
DFC404D1 8496C3FC 47A3CAF8 533B45F9 4AF43FA4 B85299B8 8839C450 B59CAEC3
48CC9BBE CD76674C 0B9079BE 9D39AE3C 800A89E0 32E77F14 3980C178 3C467639
BE5D7130 E35B7438 0803F631 8E3C61D2 FFB10FC8 3251D3B0 5404F1D2 A33ECBB5
13139B95 95B66364 28F8C916 00C4480F 276191B5 99020D57 E191DB27 5BEFF430
B1B1CEDE 676E7CE4 F46375B1
```

w =

```
0000FE5E 5E5CE52E C1767D74 B2C52ECF E15FCAC9 69D6DCF0 6A79FC44 3980801B
15C9CF8E 0617711B 46A6FCD1 5A32AA58 945E6F63 49F0FE40 0ECD9008 736FB38C
D1DDD3DB CA1BE9EE D00F5397 7BD24338 B03A962D A9CD50FD 7F5A41C2 D557639F
BF9BA20D 2FEC89E6 C60F88F6 4F8D4340 043B5C0A 209ADC6E 611EF98E 66A6CF5B
7A464ABC 6C12F6E0 B1DAA662 2798C528 3E5C4739 05FFC29B 03AA34FE D4ED8185
0D41DC5E 4FB3A7BF A10F1666 A52566B2 6D5A5789 C696A8ED 3997FCF9 DA8B1A7A
9CA85659 E65DABAD F609EB28 84F2C83C F60ECC4F 1C5060A6 655F17DE E818AFAF
8E3CBAD2 D2D15ABC 2AD3A260 D4EBBE8F 3F63A8C5 33E53092 8BC46CE3 1F3FDB10
29957C6C 65981FDF BA32B3AB
```

r_e =

```
00000023 D6506BDD E1D3C49E 92169C52 88890F3C 50259C38 7F218621 D1A3B9E6
6880FDA9 CFA2A7F7 7A15D439 5535017B
```

r_{v^*} =

```
0001C3BB 0AC68C9C 4FB4A304 B566E253 BEDFBCBC B5BD7AD1 FA42956D 27006508
0312D73A 3C4BC0B6 978C2C0D BA34D9E5 B7491E3D 2E6E0C23 8F10E14A D7E4DE25
3582D544 9981EE2E 735834C7 6BD241B8 62942844 66A49529 BDCE5EE6 E7711E99
9C08CB25 45ADF762 26C77643 D373C65F D1CD02C6 6F0CAED6 8306CE73 1497D96E
33263435 73A5BFA7 EE62DC49 7265C305 6F49BD6F DC74669C D67F656E 995B1D15
506197B1 9EC66C1F 29CE909C BAD3DFFC 2E4CCB41 74A88E6E 69B79B50 35712D9C
32E1BE9F E906C245 2FA66081 82E9A005 AB52D8EA 6CB83712 88678DF0 2291BB63
9806DECB ACB0F055 3F25BAE7 295DCFF9 D55FC2A2 B0FDF6AD 136A0539 C3DB754C
00055648 67F6AA2A C7F37EB1 4CEA3229 BB976A4D 77285682 1B6A3820 5665AD28
B1EC867F 940EB08F EA1515A2 4059F409 FEEB4D6D B2A74597 F457544B C2E0E7AB
BA73CBCB E10A1E01 1EBB912C D4C4E747 4FA3880A 2399655F
```

T =

```
35AA6B44 2F55CE26 0E96A84A AAE70D08 6A255808 848B1910 DF29EE41 66EE180E
3C6BF535 3C0A295D DBFC89F7 EA9FB726 E44C6A2E 4E926446 CF9A4AEA B77B4155
43F2430E CE5A58BA EF2F4A4A 92EADF6D 208245B4 83FCE098 0704DF43 325C900C
1D732814 67E4501C EFC453C8 7F881551 F94DE3F1 631DA3F8 9E20187B 2D6F4E0B
5FF1CE6C F3E98A1E 7D9C44B1 73868078 47A03B0E C47ED9A4 BAAFED04 8AA6E27A
B9F5A36F A2E22207 D3651DEC B3CFF319 39012F61 F066EED6 ABDC41B7 F9FFEBDC
33A79C20 19055C96 CD6C5943 419C0342 9B715969 C2E19938 D0B3632A CAAC46D6
DA9AB048 9C050DFC 9C35DA5E B211A3B9 07DAF02A 1807023A F24287AD B2A0585F
```

T' =

```
11302F73 54D6503E B7A68B19 BD9A13F6 51B016FC C485B99F DE0F66BD 167C0F3B
FDC1773B EE6D671D A0873A32 045EA7BF 9562B010 42F908DC 70B3E867 C88DAC0E
B045A29C 0C563BEB 7CCBEF1B 95D79517 C10ABDE2 CA9345BB 55219085 B06321F5
C5E99FD9 44F00C04 0ABB35D0 A3021ECC C19ADD26 B9093ECC 1D8963B3 C7644A82
49CF48AC F849051F A67631D7 F00EADD0 0BBB8941 CAEAB5F9 11E62437 906701A8
09048730 9B065900 9413FCCE 6B3BA139 F649E498 4D492728 3EB439F7 5A0BDF8F
D2CF8B84 7AA6C6E7 1B7574F1 48A18A69 421851F0 625EC66D C7BF1E21 0F80A7AC
46A65E58 A5C19077 F1E7D109 4D23EE5F 5EE433FE FF836AEC 0BD2BA4B 19A9007F
```

c_h =

```
4C51D9DB D35EED8B CBD3D7D8 C629477B 9C7A1042
```

n_T =

```
0000C450 3D3CDEE8 D1976338
```

c =

```
59D94EA3 E9AD8AA8 C1407819 FE6F411C 6B611908
```

s_v =

```
0030224A CA1F064B 494CAEA8 8D1557F8 9E37E212 3B31AE7C 182E935F 64632267
069B126C 2891E753 393FCF63 89446561 F15E872D 1AEBAAF9 AB6620D4 533D02ED
2A75DF3F 2C9C4CC2 601A6072 75B8C2E4 FA6913AE 7A2E134E F0D0231E F589958B
9437B7C5 69CF12E4 721B21FE 5FB5CA6C 62EA7D6A B42C8D70 63C144E0 E7296A66
C91DFA5A BCC4B913 A8DCD9B2 2579E7AB 0B1C004E B7214037 A1FEB235 597F8797
5B981985 EE6A13CF E1572153 B65B4C9F C3B88D86 9412C7FA 2017FDEF C1A3EC6F
```

```
C7F67D0C DF5DC48D 2B2F7338 1D6DD6BA C1354C35 FEACCDA5 1E0184D8 D98BBCA9
735C1729 E19717CA 075AFE1B 16812439 84DC93FD 78023FF1 3D2111AE 4E8C2C8D
FA036652 4F481D15 1A4955F0 0C5F87FA BCDC4FCA AFB62DDA 6C2C1EB5 367BF648
77BA9F4B A4988AB1 C37CDA9D 856241B2 AAA26AEB 62A1E654 A33BEB8A 8F6A8CC7
CAF61C2D 72DD364D 331C3A85 F8073786 F5187F00 92C15FD9 85858A04
```

s_0 =

```
00389F3F A1DCB5FE 13CB0C61 9B1832CA 0C6CFC0F 0F9E178F 762015CE 6943A11E
09482AC2 2A0A2B52 4843C092
```

s_1 =

```
00E58F41 A6E84CC4 C83157FD 7B46DC38 3DB22F5C D70753E5 865BC17D 2B93F908
FECFDB64 8A615473 5C2BB40C
```

s_e =

```
00000023 D6506BDD E1D3C49E 921AA22F FEFF9941 46635A0A B8185795 F0DD107F
5F39F279 C5F2F6AA 00FB3FEB 70D20293
```

s_v' =

```
0030224A CA20CA06 54133B44 DCC9FAFD 26FB66EE 06584D12 3171A332 2B4DA285
09C94131 5DCEDD69 793152DB 87AD3505 27299385 2DEA266F 1E57D053 7805EC87
29845715 C4F694E6 321AD591 FC18E269 48BBFD1F 1418E217 582BF73D EA6B9397
C45D1C98 C99763CD 50A17AB4 19C73F5F 27C65295 DFDA3409 C8DA4628 54DF3AEB
1C5AB84E AC1B03D9 CBE1F6B0 74AB48BF 8BB09661 789EC749 1026989C 7D51E3D0
71AE8131 A96DB0D4 7EB31519 E5992D7F 2DE7DACC D1DC5E7F 6E5BB1F8 475B12D6
8ACA9817 58851D81 47912C86 FD1D6758 0A1B1B56 7398F763 6EC114D5 06FEE171
927804E9 44CCBD66 37003A85 7A75C79B 066CB07B 0F77FD79 324FE8CC 4BFD41A8
57A23304 0A474B3C 79A6694C 720F1369 C70FB7B9 27FB7D87 1EB8DE4F 814CF842
03147D8F CDF22B43 6CF72D44 81E1DEB2 52EA2329 A5AC721B 4D21ADB2 04F45E61
387225B6 A664B686 6EB4E794 250F1692 580C2961 BCD91BF8 C8CB6C5B
```

Verification process:

t_1 =

```
00002CEC A751F4D6 C55460A0 3C0CFF37 A08E35B0 8C840023 D6506BDD E1D3C49E
921AA22F FEFF9941 46635A0A B8185795 F0DD107F 5F39F279 C5F2F6AA 00FB3FEB
70D20293
```

t_2 =

```
E58F41A6 E84CC4C8 3157FD7B 4714D77D 540C12D5 1B1EF1E7 F6D9AFF5 A066050D
DF797C19 D774892A 94F7AD1E 09482AC2 2A0A2B52 4843C092
```

T' =

```
11302F73 54D6503E B7A68B19 BD9A13F6 51B016FC C485B99F DE0F66BD 167C0F3B
FDC1773B EE6D671D A0873A32 045EA7BF 9562B010 42F908DC 70B3E867 C88DAC0E
B045A29C 0C563BFB 7CCBEF1B 95D79517 C10ABDE2 CA9345BB 55219085 B06321F5
C5E99FD9 44F00C04 0ABB35D0 A3021ECC C19ADD26 B9093ECC 1D8963B3 C7644A82
49CF48AC F849051F A67A31D7 F00EADD0 0BBB8941 CAEAB5F9 11E62437 906701A8
09048730 9B065900 9A43FCCE 6B3BA139 F649E498 4D492728 3EB439F7 5A0BDF8F
D2CF8B84 7AA6C6E7 1B7574F1 48A18A69 421851F0 625EC66D C7BF1E21 0F80A7AC
46A65E58 A5C19077 F1E7D109 4D23EE5F 5EE433FE FF836AEC 0BD2BA4B 19A9007F
```

K' =

```
A4F9D8A0 01BA8A1F F0BC8652 4F446D25 6FE30666 C77ED82B 0D380009 DACA9DD1
45BF5DE7 D3F1E2DF 23E02917 2DEBA8CB 0066415E 0B3BDE13 D6936459 58D9A421
DFC404D1 8496C3FC 47A3CAF8 533B45F9 4AF43FA4 B85299B8 8839C450 B59CAEC3
48CC9BBE CD76674C 0B9079BE 9D39AE3C 800A89E0 32E77F14 3980C178 3C467639
BE5D7130 E35B7438 0803F631 8E3C61D2 FFB10FC8 3251D3B0 5404F1D2 A33ECBB5
13139B95 95B66364 28F8C916 00C4480F 276191B5 99020D57 E191DB27 5BEFF430
B1B1CEDE 676E7CE4 F46375B1
```

E.3 Mechanism 3

Security parameter:

t = 256

Group public parameters:

G_1, G_2, G_T are constructed using the BN curve as in annex C.3 of ISO/IEC 15946-5. Let q be a large prime. G_1 is elliptic curve $E/F(q)$, where E: $y^2 = x^3 + b$. p is the order of the curve $E/F(q)$. $h = 1$ is the cofactor of $E/F(q)$. Let $k = 12$ be the embedding degree of $E/F(q)$. G_2 is the sextic twist $E'/F(q^2)$, where E': $y^2 = x^3 + b/\xi$. The total number of points in $E'/F(q^2)$ is $p(2q - p)$. The cofactor of $E'/F(q^2)$ is $2q - p$. G_T is $F(q^{12})$ where $F(q^{12}) = F(q^6)[w]/(w^2 - v)$, $F(q^6) = F(q^2)[v]/(v^3 - \xi)$, and $F(q^2) = F(q)[u]/(u^2 - \beta)$.

In the following numeric example, $P = (P.x, P.y)$ in G_1 is represented as $P.x \,||\, P.y$, where $P.x$ and $P.y$ are elements in $F(q)$. $r = (r_1, r_2)$ in $F(q^2)$ is represented as $r_1 \,||\, r_2$, where r_1 and r_2 are elements in $F(q)$. $s = (s_1, s_2, s_3)$ in $F(q^6)$ is represented as $s_1 \,||\, s_2 \,||\, s_3$, where s_1, s_2, and s_3 are elements in $F(q^2)$. $t = (t_1, t_2)$ in $F(q^{12})$ is represented as $t_1 \,||\, t_2$, where t_1 and t_2 are elements in $F(q^6)$. $Q = (Q.x, Q.y)$ in G_2 is represented as $Q.x \,||\, Q.y$, where $Q.x$ and $Q.y$ are elements in $F(q^2)$.

p =
FFFFFFFF FFFCF0CD 46E5F25E EE71A49E 0CDC65FB 1299921A F62D536C D10B500D

q =
FFFFFFFF FFFCF0CD 46E5F25E EE71A49F 0CDC65FB 12980A82 D3292DDB AED33013

b =
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000003

β =
FFFFFFFF FFFCF0CD 46E5F25E EE71A49F 0CDC65FB 12980A82 D3292DDB AED33012

ξ =
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000002
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001

P_1 =
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000002

P_2 =
E20171C5 4AA3DA05 21670413 743CCF22 D25D5268 3D32470E F6021343 BF282394
592D1EF6 53A85A80 46CCDC25 4FBB5656 43433BF6 289653E2 7DF7B212 BAA189BE
AE60A4E7 51FF0350 C621E703 312826BD 55E8B59A 4D916838 414DB822 DD2335AE
1AB442F9 89AE5AD F80274F8 7645E253 2CDC6181 9093D613 2C90FE89 51B92421

SHA-512 is used as the underlying hash function. Special hash functions are realized using the primitives in Annex B. Optimal Ate pairing is used as the underlying bilinear map function.

Group public key:

Q_1 =
13B9155C DFDAAA36 2264EDC4 CDCCEE31 62D30ADB E7560ADC B79AE0EA 789E0197
8CD127DF 3D171922 C96D8A19 C0F1E043 FB40E88D 5A3D205B EE0075E3 3EBD7BB1

Q_2 =
ED1BBE1F 064A969A 34DD5193 0B308272 B71D1066 B9F01F0E 960BBBC1 CB89C72C
EC62B0B7 E760884F 99E07FED 478EE679 2E90E531 78D7A334 2004D3AC 906771B0

W =
DFE0F697 65D4B585 32E96F63 BC4F89B1 6B45A77F 877B6CDC 33B42129 148F0A0B
A2B632CA 53E96E80 02B51788 F209285D 1DF7CE67 8AA5AB37 BFD9767E D9B5CDFA
F31DEA38 6566E809 8D08E07F 1F3E2B13 A081442A 034CDAB6 68408D80 FE8A0E85

D48059DC 02B7243F 77650281 B8527C14 0249C560 9588D762 23BC464A CA24062D

T_1 =
A88E9AF9 251298E2 C3612EE8 D6A67716 49047569 D1832D3F 2A79B69B C91D0390
2AD8119F 2636E7E9 3A054C15 4993DAE9 D05AE48D 8AFA04F1 208456EC 3C27195C
F1AFBFF6 0E58842D 9411F4B5 F41451B0 90461A81 EDCF9166 58A6363A 52185AC1
084C99D3 DCCE7FCE 78E03887 32F1803C 7B67AA6F DDE0FCCB D0B03A59 522A84E4
F84AFF50 A065C4EE F49CAA34 46F9D26C A1617149 32258454 9044BEA4 0BF7FE26
816373F7 2FF2FA24 52A4D94C C1A7A5C3 0336139B 164516CB 4B9938F3 6DC87EAB
B353DFB6 82601211 36690E05 318ECFD7 3F32E795 841DC8B5 BE49179D CFA95A2A
C41186E8 6C0256B0 252FA006 B362B211 AFBEA4E8 616485FB EB1CF1BC 2CAE1051
16A6C0B3 868E6D79 B6BDDE1E 26064665 82845A97 D3B79378 6B9D1433 94433404
45D147D4 2F17CFF1 DDEA1152 AE01883A 10EE5C16 CDB548E9 162C70B4 1E1938E0
18E9AEC5 DA74412D 70076037 2766F700 BB7951F3 7C8A2BB5 696E101F E00A5EBE
B44E0E02 59B5CB4A 6A868BCC A213A0E9 F25CB023 B215F9BB 43C154F4 C8AB16A6

T_2 =
EA09DB32 1E161D34 D8AE05D0 E6440E78 175E33D8 05A8A979 3D4D3628 997E0794
696E6F2A 99D45E05 F7F84CC8 5749ED15 24E72483 128EE61E 44BCEEB7 901C81AC
B245C825 8808DA12 0DE7D923 0E0E7AB7 A718B25E 1C620095 4304119E 8DD9B5CC
0EC8AE34 766D6CE9 EED9C35C 913BF832 EA0B7D5B 558DCB73 91B4F060 1124A223
BC52CBAA D25F9B10 E9BC87C4 8C1A0BB9 75C31AF0 2C29ACFF 45C14AED 7D7846B3
CDB7E64A 917DDC27 783AE81D 28AFAF40 CC6ADE60 12E11C9C 21AE1854 4335410B
5D148C3E 1D397334 437CD6A8 53E692AD 354484CA 87EF86E6 A4A594F1 D9C264D6
B9061BC8 73B0C390 0005B629 31C9FC1C D48F78B5 0BBEB79D 0ED8C2D7 E59D6BFD
8F6B9F3B D37FEB6B 859CF998 FDC77376 87839B01 56658267 5C062664 F0B1CC4C
5F7B3D4B 86474FDE A6EEBED3 362FE562 66B50DA9 B4193E17 CD2E0726 59F1556B
97C6559E 099E05B7 618770B6 EA75498F 016A06E4 5255B296 88C727CF 59AE9B8A
94763F4B F89C4615 23DB6FCD 210FDE6B ADC58BCB 48433AD2 F4B7C5CE A2B311ED

T_3 =
77D07370 3278E443 F1F61001 A0E1534C 833E9493 37D37F1F 6D627BE8 252B9212
6C176BC5 2C8223CB FE6CCBD5 A32F0591 62FC0882 5DB7E2A4 65B25F37 4285E6D1
A20C6F53 4720821E 9F3C6111 08FDB364 E7A53A32 AD67C886 E9B8F822 66812D87
B190FA69 0B5E29D6 9A4A3329 C3DFECB8 0E77C08B A66B1DF1 C3981DFB 2728D030
B3E0E570 22DBF9A1 93C67759 237CCEDC 44B68C83 0C3EC75E B60B08CE 8391F3B6
12518762 34B202F9 89DE6287 93488CEA AFDB133F 4902C350 6398AD0C A1DCE367
5DDBA3C6 9C7C32DC C785BBE6 B70762BD 3C50A989 82E6AF3D C9E69F5F BAF48BA6
910EFA01 2AB49848 5C8C0DE2 F44F04E1 F35CF786 1FF5D220 D6A37EA0 8C3D867E
642B8033 5581AD42 F9E0DA36 EE1196FF DACF211D 6DF20422 4131EE3C AA8B54B9
D61E662D A62D458A BFCE38AE 24899DB0 87B37DE8 63EBD2A3 9F06431C E0E71680
98A3D7AF EA21574D 7B806D1B E6EA99EB F4D96589 DFE34CE2 CA605E44 71789970
77E0752A CB248031 F43CDFD9 C2B82A28 88DE9DC5 1E457D71 E4BDC999 A518B75F

T_4 =
10306554 CA58B236 43AE81B7 0DF214EF 741D822F 443076A1 DF8F266F 1EF8053B
21F605E4 C9C4B77D B33D4091 26440477 F147ACC0 66BB02A6 5ADBCCA2 8DA5A4A3
8EE50EAE 0906020D FF213F98 C8157452 474D8074 8AFEA3C9 86B7C879 A854FE7E
1C97B19F 3215C4CC 4AE5C058 83999BDC D75DE341 E40EB4B9 189BD1F7 9EB777E3
503522BE F06CE9B0 DB028D36 A101A4F3 9ED6524C 74916018 EBB2D391 EB709F90
3127658E D21D5E2E C79EDB84 153274B5 7C9F8E30 9342F54D 25966FCC D5B65645
3DA22F68 BEDD702A 8EDD2439 95B3345C 3DB9CCDB D805C1D1 67F08CE2 4F52108C
D16D4D75 476DC0E4 412353CB DA41FB8B BD6991E3 62AE1954 6CE9DFE8 79E6A669
DB9EACDE E9DFF5F2 97145994 7857F84D 43C8C9A1 EA2AE009 E571606A 4F66295A
3D050A3C AA4E4E59 12AE1781 33F816EE 67EAF1D3 2D129ADD 3C382CD8 95CB163A
6E5AFE63 21D17C65 48D42905 444F8736 B8F2605A 5CA9496D 4FACB65A C9E50162
750A7FDD 1D3583C0 D44541F8 3ECF1FEF EC1E435D D4573418 82638390 D7A61B66

Group membership issuing key:

y =
4484A7A0 B04BD4C6 A5D22451 96BFCCED F0A256E6 15C05E70 29C1058A 5B5EF4A7

Group member signature key:

f =
59A5DC1F BEEED102 D3D17F7D D6BDD1B3 B7C2471D 7E824781 477F9297 E9F1ED58

A =
99F75E90 879DD188 C6866E61 85F6C5FD 8EB9FC68 4DFC40C5 36C8EF7E A7FB11F0
700E5ECE EB64A2CE A0DC73F7 379E5B80 39443B70 0C9E5DA6 EE5E0614 A784C150

x =
9EF27558 0B86DF21 30E65A8B 10CE5AED CA8D285E 5665BEA4 D7A18DCB 14FD26FC

Group membership issuing process:

n_I =
0D5E27F6 7FD0BAE7 B4558495 0B7E9241 C526897B 483912F9 E163D41B 760CD838

f =
B1B1CAC2 285AD778 7BE67201 37311A75 5BA167A2 367FE4DF 39A9A119 DBEA455F

r =
B35ABB63 E0143D81 254A87D5 29E6B4D1 1466F1C5 AF6E30D9 0E8B5C92 4CF6E1A3

F =
57E03F20 F9A9E164 0B5E7DD3 FCA4E8EA 71CBFE81 C63C7347 B8C4A102 345[illegible]B6A
4DE234F6 0A86262A 25B7F1A6 D7D24C6B A282C04A F175F935 1E9702AB [illegible]458EF

R =
BAADC096 476533E3 3A81D123 D1C90281 35B71732 D0573597 D5C22656 BCA5E71D
47B50B1D 02833BE2 A1480713 8E0BECA8 90678107 683C5920 28108D75 EBC07CF2

c =
A1667B92 7F17EF38 1498A4E7 FAC2397B D6D12CB4 DAAD59B1 1F9BCB9B E5E8871A

s =
1B187D33 E4CE10C9 0C4AA6E5 EF5742D5 C0B4BA7A A743144E 7427C90A DC8A19AD

R =
BAADC096 476533E3 3A81D123 D1C90281 35B71732 D0573597 D5C22656 BCA5E71D
47B50B1D 02833BE2 A1480713 8E0BECA8 90678107 683C5920 28108D75 EBC07CF2

c =
A1667B92 7F17EF38 1498A4E7 FAC2397B D6D12CB4 DAAD59B1 1F9BCB9B E5E8871A

x =
263FF7FA 973F78E5 51863EC8 F892FBA0 71662D5A 0355C214 422BC4ED CA0D21FF

A =
DCF4F117 DCDAF1E9 187[illegible]BE 9DE95870 091E0C78 9D3C6149 A8DAF221 F37F8FEE
D5B731A3 693C0F17 D14D05F2 8F59605A 0872A1D3 D2055801 DE02BB19 5A7823C2

Signature process:

bsn = NULL

Message *m* =
0460EAAF 6D8324EC 4D04EC06 42B1EE33 BDC66299 9C39453A 6CE7AB32 CA00EAB1
ED60AA91 4125EBDF 3846C71C 26EE64F7 D49BFAF0 510F2131 87728A83 C639BFDC

J =
088AF78B 3EF3F55C C0CB3D6E 29E7DA58 2EA35EFF 8DAC4556 45509925 73E40926
3AC4611D 126147BE 06632127 05B7196D BEC0DFB9 4C2363FE 784D2236 C8D7064D

K =
B3187052 CB49EA57 AD88018B 2D763FAB C4FAF474 9EEDD937 8D351D5A BCB69A61
2BA4EA00 2F09B0AC 18943F24 A8DDD28A D9B6652C 4EC77F4D 605BFD7C 056BCAAE

a =
BC907239 6207C3EE 7D7963DC 557D5C48 B421BA83 63F7EEEC 1ADC91EC DBC20B87

b =
1CE910C9 2A6E6089 6FFFDF5D 6C6A539A 43E8B9AA 2CC2D8F3 777A4D38 BCEC38BF

T =
99C83125 70E7B0FA AEDA0C52 2DB86180 B47B9ED0 0F522BEE 77231F78 F7E0B877
EA63DE9B 901D0628 3F12CE61 E82C4889 DFC14885 746CAB3A F03B3AE2 86D0AF4C

r_f =
5D6654EA 623F76D4 C66698C6 445E5052 738A0D4F 67282065 428A20B5 75462809

r_x =
BBED75EE BD0E36CA 68AB0381 C0B7FEE4 5CE8DD1B B76D7DDB C4E3A317 17DC9784

r_a =
D4C0E43D EBD219CC 31FCC123 A5170CEE 05B88386 9861081E 82CDD950 2EE7ACB4

r_b =
93D2970F 7816C3ED 4B431626 76613E4F 764518F3 977E5A51 14D4943D F9394A81

R_1 =
F2AE4446 6A998700 F64E628C 877F58FC 71260C63 CBA058FF DB3069FF 2E0328C3
00DD2A5B F9D0EB4B AC25C9F5 09F86F02 7E97B0F9 FF86635E 0776B634 726BCBD8

R_2 =
4B80317F 1F383E10 3248CAF8 C9F17415 9DE322E0 0B1312D1 7195350A 9EF3CEA2
6F3E0F3C 421F7B87 D51BB83E 3ED9D425 81C66598 1BCBFEB1 9FE7A830 70A24A50
42BB1F77 7477C95A 78247A5A 1F559C69 93C2C188 CBC8BC1D 8217E418 622CC8F0
16964584 AF3F46D5 B888FE98 C2F23DCA BAD76801 7C31FED2 DDFCC8C0 6154DDA1
565133D3 FA53406E 254A42C3 7DB02DA9 E582B8A6 8E9C13BF 895DC044 8A39DBF6
C8ED4EB6 2E62314E 43D2D2BA 48EBACE6 5F1B911B 1E0FC5EE C052F41A 024C71E7
DD56A764 1F50BB11 809EA9E3 3D0C564F 538A924B 228AE287 E936F368 695E392F
E4FB0577 DA1381B4 7427BA11 E5FCC0B3 4A77777A E75E08B8 9AE35C66 279E925E
5959C456 885E60AA E91CCEA7 F9BCBCDD A6552B89 28A5CBB6 26C319A8 23710080
3C881E86 DB8BE86B E1F0DE10 CF95924E 9CCF3880 09F1C9F0 BB961783 E78691BE
0F8272ED CD936ABB FB8B9EAE 29A685F9 815F0A9D 4682F299 9B027024 8EB9A659
C6F4728B 31FC3E16 3717A259 F6214594 D14A3FC7 E5BFAF64 93AFE578 ACE8CA34

c =
CBA0D828 1B49750B 3AE9601F DDF83481 CA848EE5 2150A9C8 CA189F44 BF7559E0

s_f =
43CF67A4 80EDE9B6 26327E7E 3E59D4C2 C1D80BF1 68817C32 6E4BF694 F8C5E940

s_x =
A7C8EE59 BE1C511D 45FFE1AB DD37D978 A4119EC7 ADDBB5A2 B7F693A8 040A1CF5

s_a =
4962D9B7 29F06A62 60841392 9530F407 A94341EB 223831FA 67483666 3BF5AAF4

s_b =
1F00F7F0 F7C4C0AD 400A5281 51BB6DBB F7FF24A3 78C0BCE3 21FAFD8F 76CACC1D

Verification process:

R_1 =
F2AE4446 6A998700 F64E628C 877F58FC 71260C63 CBA058FF DB3069FF 2E0328C3
00DD2A5B F9D0EB4B AC25C9F5 09F86F02 7E97B0F9 FF86635E 0776B634 726BCBD8

R_2 =
4B80317F 1F383E10 3248CAF8 C9F17415 9DE322E0 0B1312D1 7195350A 9EF3CEA2
6F3E0F3C 421F7B87 D51BB83E 3ED9D425 81C66598 1BCBFEB1 9FE7A830 70A24A50
42BB1F77 7477C95A 78247A5A 1F559C69 93C2C188 CBC8BC1D 8217E418 622CC8F0
16964584 AF3F46D5 B888FE98 C2F23DCA BAD76801 7C31FED2 DDFCC8C0 6154DDA1
565133D3 FA53406E 254A42C3 7DB02DA9 E582B8A6 8E9C13BF 895DC044 8A39DBF6
C8ED4EB6 2E62314E 43D2D2BA 48EBACE6 5F1B911B 1E0FC5EE C052F41A 024C71E7
DD56A764 1F50BB11 809EA9E3 3D0C564F 538A924B 228AE287 E936F368 695E392F
E4FB0577 DA1381B4 7427BA11 E5FCC0B3 4A77777A E75E08B8 9AE35C66 279E925E
5959C456 885E60AA E91CCEA7 F9BCBCDD A6552B89 28A5CBB6 26C319A8 23710080
3C881E86 DB8BE86B E1F0DE10 CF95924E 9CCF3880 09F1C9F0 BB961783 E78691BE
0F8272ED CD936ABB FB8B9EAE 29A685F9 815F0A9D 4682F299 9B027024 8EB9A659
C6F4728B 31FC3E16 3717A259 F6214594 D14A3FC7 E5BFAF64 93AFE578 ACE8CA34

c =
CBA0D828 1B49750B 3AE9601F DDF83481 CA848EE5 2150A9C8 CA189F44 BF7559E0

Split signature process:

bsn = NULL

Message m =
0460EAEF 6D8324EC 4D04EC06 42B1EE33 BDC66299 9C39453A 6CE7AB32 CA00EAB1
ED60AA91 4125EBDF 3846C71C 26EE64F7 D49BFAF0 510F2131 87728A83 C639BFDC

J =
1A0A7ED3 F9381C38 FB34ACE5 AB735AF7 9E35E06F 88EDE2FA EE34B595 249C06C1
BC0C71D7 F707890A 4F39409C F5FD6A0B A00CC0F1 4B432C85 EA44BF39 A842093B

K =
C3CB5651 BB35B629 3629E0FB 856FF712 9705D791 D7333CE1 8858C4F1 1ED50703
2D49F1DC 0B3F68C9 40EB9579 C02C6702 41127E04 3AD710AC 5726A949 56F04BD0

r_f =
7E30ECD8 86B53233 EF7403B4 8CEA199F E950B959 A6EEDA7F 87DD2350 A482F637

R_1 =
9A70F54F DCC81B2D D4D67683 678E1219 7B2E72DC 5ACBDAF4 51C6C547 80AB7341
041E9FDE 4FF5E949 D9DBCFF7 CF33C2BD 72809B60 AA63E4A6 2C634181 0C2D871D

R_{2t} =
604FCC1B AA861786 F747635D 471C69E6 9EA5C45D F6DA2E1D 1E8DDF41 4854DB2C
4EF8CF5F 1BD5EC1B D13EC007 A5740FED DA8152C1 0521D2C3 A004C1BD D72057BB

a =
742B0A9F 38F23E74 761205E9 79059E6B 0FE9503E EF75B0FA 2C5EA463 C12422E

b =
2991FA0B 4845CAF3 1F9ECFB5 9D0286C6 0266E7DE BA8351CF 1921C523 A6ED716B

T =
79CD7C12 E7674B03 51FADD06 4BC439DB FEBAD3E3 4D656D68 725404B1 58D03420
B82538CE 93C37B50 6652F862 DD172E10 E5F5A6B3 C4ED20E2 509B8F0B EEF147E8

r_x =
11547E69 7DAF14E4 416E0649 2A333385 40D07E84 07A25821 A42E96E2 FC535DC0

r_a =
8B0609B6 0A131D79 AFEF21D8 03E5597D 4B0BDE83 82EBF786 EA023E77 3A9EDA5B

r_b =
7A2A8F3B CD46D6A4 DB1A9F2D 25EBF959 01AAAA53 8E88B7E3 A1D2CF79 6718728F

R_2 =
634F27DA FD3BE93F F88E3D28 1A0E376A 55222888 D05DA2FB 79A31457 690874E5
9E619740 821CD9AC 21F1D71C A0BB1D64 9CEC3EA2 E587F2BF 99E2B8F6 1311CF9F
2AE033C1 BC3416DB FF2C2EC6 92BFB870 BCB89126 7101F018 D61E167B A644DF25
425D17DD 7B7005E5 D5D4735B 5D5AA4AC A190150A 7DFB8687 AFB229D5 3FF98468
38C3B29F 3144AD67 EE0A1ED8 EABD3F2B 1B41B03E 64871946 44FCA932 25DA5312
FF5972BB 5054DDB2 74E96E56 E2593F3D F440D528 68B2F45A 7E812A31 053035E1
5959E583 1C9CA236 1D370850 775E2671 858852CD 7E88FA76 2C2471B7 DB30AC27
C8D3E2E6 A59C2F5E 4881263D 993962C2 599305E6 8DD106F6 F005F6AC 40174DE0
0274B4FC 8B3D00F9 864D6434 36045EDC 4512A208 F1F5A603 309924E7 0387E168
A0295A19 280C1D29 4681E35B 218B50E0 78B67D84 906C143D A69D35A4 1F4254A9
33506472 35051018 BDFB6E31 8C0620A4 1AC0C0D8 476F3DC7 FA901DAF E0D92427
69920899 D653D7DC 700B0584 FD8DAC53 32485439 B94184DB 248DDE44 0868ED9D

c_h =
434247A4 4663A599 08F1635F 4E292238 2A234CC0 735FB2EE C013EE8F BEFCC643

n_T =
8B46B532 D32CA5B7 1774B0D2 006A4217 513E6697 02146A31 A83D7585 0642E9E7

c =

A2F2CA51 EDE93FF9 188AB125 D9B82C02 02C4C3CC B0DC8905 CDE1FA30 95B985F4

s_f =
9EC76930 DF5B05E7 59408FE6 A691F8F5 DAE31E1E A2340CF2 0F49982F FECFCC14

s_x =
85953B61 5415407E F7CCF6DE F784EC60 111309DD A99C776C 5BBB77AF 4FBE23D2

s_a =
3A4E1602 1F574809 063E2FDE BEEC154A D7FA13ED 0C4C95C7 86DD27E4 B58339F2

s_b =
48B5E17D A8D835F8 DFE64023 93009BA9 0BA25D4B B5FEE9C2 F7354BD8 2AC3940E

Signature revocation:

J =
088AF78B 3EF3F55C C0CB3D6E 29E7DA58 2EA35EFF 8DAC4556 45509925 73E40926
3AC4611D 126147BE 06632127 05B7196D BEC0DFB9 4C2363FE 784D2236 C8D7064D

K =
B3187052 CB49EA57 AD88018B 2D763FAB C4FAF474 9EEDD937 8D351D5A BCB69A61
2BA4EA00 2F09B0AC 18943F24 A8DDD28A D9B6652C 4EC77F4D 605BFD7C 056BCAAE

J' =
2E6A75DF 72B54FEC 6F8EC765 D32CF139 80720BF6 E39418B1 0A69AA30 BB983C1A
82E892A7 8AE32B6C 7C9446E0 FD02DB98 9165E830 9C8438A6 5DC77A34 6279C95D

K' =
F70B4979 3D37BCF8 88B9BC33 0C44037E 1FF02AF3 F660A110 ECC0FD99 823B5C70
A805C2C2 72C34FED 7586AA0E A06B6A2C 2A3DFE9B C97CF6F1 F1418AD4 7CFC1C21

Message m =
0460EAEF 6D8324EC 4D04EC06 42B1EE33 BDC66299 9C39453A 6CE7AB32 CA00EAB1
ED60AA91 4125EBDF 3846C71C 26EE64F7 D49BFAF0 510F2131 87728A83 C639BFDC

u =
C2696A55 90152F1A F0B097BE 5BA38764 95D18C44 E34C96E6 DE6AF6D4 E6D6C845

v =
A1ED15C0 83ED601C B641CEA8 60940FC7 6C428B7B 5921A76F 9CC8784A 6ABE14D3

T =
A65D5F0E 896F767F 386CC5BC 1F5E24A2 C5390881 72768A82 A7606B71 A03EFF28
432237D2 4FCA8C2A 90B6AE11 DA512AC2 61907813 A65F4987 C5E66275 FADF6C8C

r_u =
4123691F 44DDAF84 B07B74B3 79355352 D31BF942 B967DE90 E8E7407C A49A0BD9

r_v =
B3082FAF E4E27122 3CDB2148 9B5424AD CD393739 1C55ADD7 4E637C6E 680EEBFB

R_1 =
9A4EB3A1 5AED2CBF 8C706159 58C2936B 400A81AB CD53C007 399BFFF7 3429ABF6
E5B2AF58 D2F70AD7 8C551940 50FB7971 5FB607EB C919703D 55C9C782 33285D25

R_3 =
2E34680B EDE498D6 6BE819A7 FEE2877F 0D296C9D 829590BA 472604DB EC2E1F6A
1EA38AEF C0BB61C0 91151DF5 1953705D F5BE1131 02744B90 E4955BFD D8BC13C5

c =
5FC97235 909666BC 21EDDEE1 A7BFA1D2 A965D936 63B49BC8 FA456F1C B150028E

s_u =
7031B184 779CCFA1 FCCB306C 5E001530 003E2299 CC26AA85 C5D949A3 742E2AB2

s_v =
DE394CFC 2325071E A86D33FD D7314643 2E6069E5 D4D6742C 111C448C E37A19D6

R_1 =

```
9A4EB3A1 5AED2CBF 8C706159 58C2936B 400A81AB CD53C007 399BFFF7 3429ABF6
E5B2AF58 D2F70AD7 8C551940 50FB7971 5FB607EB C919703D 55C9C782 33285D25

R3 =
2E34680B EDE498D6 6BE819A7 FEE2877F 0D296C9D 829590BA 472604DB EC2E1F6A
1EA38AEF C0BB61C0 91151DF5 1953705D F5BE1131 02744B90 E4955BFD D8BC13C5
```

E.4 Mechanism 4

Security parameter:

$t = 256$

Group public parameters:

G_1, G_2, G_T are constructed using the BN curve as in annex C.3 of ISO/IEC 15946-5. Let q be a large prime. G_1 is elliptic curve $E/F(q)$, where E: $y^2 = x^3 + b$. p is the order of the curve $E/F(q)$. $h = 1$ is the cofactor of $E/F(q)$. Let $k = 12$ be the embedding degree of $E/F(q)$. G_2 is the sextic twist $E'/F(q^2)$, where E': $y^2 = x^3 + b/\xi$. The total number of points in $E'/F(q^2)$ is $p(2q - p)$. The cofactor of $E'/F(q^2)$ is $2q - p$. G_T is $F(q^{12})$ where $F(q^{12}) = F(q^6)[w]/(w^2 - v)$, $F(q^6) = F(q^2)[v]/(v^3 - \xi)$, and $F(q^2) = F(q)[u]/(u^2 - \beta)$.

In the following numeric example, $P = (P.x, P.y)$ in G_1 is represented as $P.x \,||\, P.y$, where $P.x$ and $P.y$ are elements in $F(q)$. $r = (r_1, r_2)$ in $F(q^2)$ is represented as $r_1 \,||\, r_2$, where r_1 and r_2 are elements in $F(q^2)$. $s = (s_1, s_2, s_3)$ in $F(q^6)$ is represented as $s_1 \,||\, s_2 \,||\, s_3$, where s_1, s_2, and s_3 are elements in $F(q^2)$. $t = (t_1, t_2)$ in $F(q^{12})$ is represented as $t_1 \,||\, t_2$, where t_1 and t_2 are elements in $F(q^6)$. $Q = (Q.x, Q.y)$ in G_2 is represented as $Q.x \,||\, Q.y$, where $Q.x$ and $Q.y$ are elements in $F(q^2)$.

```
p =
FFFFFFFF FFFCF0CD 46E5F25E EE71A49E 0CDC65FB 1299921A F62D536C D10B500D

q =
FFFFFFFF FFFCF0CD 46E5F25E EE71A49F 0CDC65FB 12980A82 D3292DDB AED33013

b =
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000003

β =
FFFFFFFF FFFCF0CD 46E5F25E EE71A49F 0CDC65FB 12980A82 D3292DDB AED33012

ξ =
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000002
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001

P1 =
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000002

P2 =
E20171C5 4AA3DA05 21670413 743CCF22 D25D5268 3D32470E F6021343 BF282394
592D1EF6 53A85A80 46CCDC25 4FBB5656 43433BF6 289653E2 7DF7B212 BAA189BE
AE60A4E7 51FFD350 C621E703 312826BD 55E8B59A 4D916838 414DB822 DD2335AE
1AB442F9 89AFE5AD F80274F8 7645E253 2CDC6181 9093D613 2C90FE89 51B92421
```

SHA-512 is used as the underlying hash function. Special hash functions are realized using the primitives in Annex B. Optimal Ate pairing is used as the underlying bilinear map function.

Group public key:

```
X =
81ECB895 667EA4F9 F37193F1 EE91968D 0E1677D8 42C9D98C 0731486 D1797A492
0F31D669 D93543F9 23484F76 3EB07485 EAD88D90 EB277476 7F4A599 00253F849
FF83F12E 98791CA7 63A900A8 94CF2690 6E42CAB4 E96B614D 2E2F468 1B7B5D1B1
BC97D3BD F100EC4B 16635FA0 3B4959B5 58ADEF4D BE6D8904 0CFC739 9A294195F

Y =
```

```
A7F6DBE3 D5FE924C 92B87B9C 87D25132 FB464A8B 48032A70 DFD4844 B588FE585
504147A8 64F90C5C B22C49D3 2B9357CA 51760D52 621CB632 50D522E AAB9BB271
0910BEEA 0B55068B EAE74888 75A02E51 46B37C9C DEC6B2B7 C74FCA2 9E2ED2AAB
4E148283 F3E99483 8A24F2C6 903EE6BD E99EEFED F2D137F6 3BDED47 BE46297A8
```

Group membership issuing key:

```
x =
65A9BF91 AC883237 9FF04DD2 C6DEF16D 48A56BE2 44F6E192 74E9788 1A776543C

y =
126F7425 8BB0CECA 2AE7522C 51825F98 0549EC1E F24F81D1 89D17E38 F1773B56
```

Group membership issuing process:

```
n_I =
D2815256 86B5897A 1396AA5D E0509543 E319E3EF 3EC1BAB2 24CDE6E9 5BE18CDA

f =
05E8D2E3 F942A58F 652CE4B7 2836BB01 23AF440F E74004CC 0E0F37F5 59BAC367

Q_2 =
2F858C21 7C1F2818 F1912A72 20852462 8AE6FC53 49A97D82 D6ACB646 AD3A4284
B1A886C3 3E5443AF 1499EF32 F0CB5186 B7F25E52 FBA05426 CFD590B1 974143DF

u =
84A7A234 A62153E1 158405C5 C8A64EC6 6ABA6220 CA230421 460C3F4C ABF83879

U =
1B740B6C 6D316705 8193305C 5D9E744D EC93493A A9A28539 60410205 590E7CC2
49038FC8 4B9BBC70 39330E1D 6A01F156 8D537112 CFB566BC 5EB6F470 8F83134F

v =
D4338FA7 9437AC22 04DE4799 4D0D728B 9745DB69 74FAF25E AC7409B3 8E562067

w =
E37A4E0C FAF2ECDD C3F99A21 D7D293D4 C3E30982 AF0DD17B 0E6494E0 5F4721AD

U' =
1B740B6C 6D316705 8193305C 5D9E744D EC93493A A9A28539 60410205 590E7CC2
49038FC8 4B9BBC70 39330E1D 6A01F156 8D537112 CFB566BC 5EB6F470 8F83134F

v' =
D4338FA7 9437AC22 04DE4799 4D0D728B 9745DB69 74FAF25E AC7409B3 8E562067

r =
7FD81AC7 9FB82ED5 003ACD4C 0FB73365 CA30CABA 724AA5FB 4FBBBDBB A708F736

A =
8280EA5B 44404EA4 1468AE3A 3273793C EA5A03B6 88DDFA1E 14E47244 2B45ED8C
49DF9DD8 350C7294 AC9AF0B0 8608969D 4CA0CE4A F62C7A23 E87BF770 3EBDAD8E

B =
DE8C62EA 37A647AF 3450AB3D 88219603 80A0700C D4BEFF66 E7B33986 D4399D15
FED6B937 580468CD CF862F71 F12936E3 261EA0C4 0FE2C24A 5A50BC51 9D94D039

C =
6A8DF72A 3D2E7AF9 C8D68FFB EA8174A7 BA44A04C 39D2E72E 6A8ED491 390E5314
8DBEB136 02EEB668 11E75FE0 DC9F856D FAF990C6 6B10778A C23078CC 19C7FA1E

D =
B4C3017D FA57BA15 7B738B57 3C52F553 70AAC3A5 3FCE4132 4C13DDBF 2F54DF96
08B03522 315970C5 A9493A07 7C57C6EE B11AB863 834554E2 6DA1B10D 87053A8B
```

Signature process:

bsn = NULL

n_V =
F7BACAF1 4B704BCE C0CFC1DA C078EA18 94D2FEE8 ECD79D96 472588E0 EF1DD3FA

Message *m* =
BD06E6EB D43C94EC C8532643 93825074 FE1BA5F4 E59DE3F8 38E52146 F90C925F
EF39BA7D 98AAFD8F 07426D7C 3F5E39D8 1B7B358D EB330205 71C311D1 494E430C

J =
525998AA 0F1375C5 675CD264 66B2191A 6CDF0ADD 4B8D815B 27621253 1BC12924
4CB95CD2 AFC76DF1 31BE43D3 DF81866E 0E90BFD2 816E8D00 C434BE4E 4A41AB0D

l =
D954D9B8 C3168F2D 926D925F 8675FB1C 34BA0C67 45D472C7 0907571B 0E6460C3

R =
83AF9C12 8B18FF37 CA710FBA 5B67DE20 10786545 A275AE4B 26BE10F8 F83A3EE1
624093B2 67186E40 FFCF788E EE7F538A 9A9FBE61 F7E74ABC 6B6FBA20 CCD71FA8

S =
6B72A84C 752AE794 0470B90A 497D4EEB DF931490 9061635C F26B3D57 69D62B81
859DB022 EBDC2CB2 D9EA337C 9879E647 83ED4EE7 0D5122CB 45B869D8 6A078112

T =
9E02290E 7681DFB3 2D6DACFF 7A7EF05A EA6D4915 A260BD0D 449C3123 BB844524
411E058A 1184E12C C97BF7D3 A51D3FD1 94553E54 B0DCB4E0 EC1B17BB 7E7F2457

W =
2D476F4F A32726E6 55CE2338 AF6D4C74 FB53CBFB E70D3089 4CB9D51A A435B836
86EB2076 AA7EDAB8 22AA7AFC FB87A21E F34B4266 B83A1A93 CD405BA3 EE3A4A5B

c =
C0B08FEC 2BEE0485 DF041E36 8510FB80 0388AE8C 6CF582F2 7CB36FC1 F69FA594

K =
C8C20A5B 4556E0B3 02224900 4BCBF8CD 7B0F6D0E D06801E1 E4835E1F C3226538
60EC31A5 D75EE611 04EB3A11 4C6D58B8 B7532E26 4C2F58D2 5B43C479 6979F7C2

n_T =
38C2842F F986CDF6 BE7331F7 7A8FFCD0 7810CC43 CC07CA27 7B33B109 7D3B685D

r =
DDC650F8 29907BA5 1B255597 BF83ECE2 183A8D6E 3A2AD054 DBA6A7D9 7AC5AEE8

R_1 =
E57877BE F012310D 15C5EDD6 32B747E4 DB9FFD06 A3539485 0419DC2B 8B456093
D80DA3FB DD5A4EE6 2F80C4D2 5CD69407 477C350E C64022D3 195FDEA3 38B6A1D1

R_2 =
ED9A846A 104EB7C5 0361C67B 2FB92A05 8CD245C7 ED9F172B EB449F41 3D675CD4
86B17A09 A511948D 09F47D96 2D36E4ED D8BEFF84 0BBE9515 8D55C98F C65716C8

h =
B329C912 53094139 4DFC1162 043A458A D9CC21C8 4F08F22C 6BB480B5 362D8F89

s =
E3D906C7 D2D5D0B4 BC13ED90 DFDC26B4 AD86C66E C9793898 F54A3305 0BE591D6

Verification process:

R_1 =
E57877BE F012310D 15C5EDD6 32B747E4 DB9FFD06 A3539485 0419DC2B 8B456093
D80DA3FB DD5A4EE6 2F80C4D2 5CD69407 477C350E C64022D3 195FDEA3 38B6A1D1

R_2 =
ED9A846A 104EB7C5 0361C67B 2FB92A05 8CD245C7 ED9F172B EB449F41 3D675CD4
86B17A09 A511948D 09F47D96 2D36E4ED D8BEFF84 0BBE9515 8D55C98F C65716C8