INTERNATIONAL STANDARD

ISO/IEC 20008-2

First edition
2013-11-15

AMENDMENT 2
2023-04

Information technology — Security techniques — Anonymous digital signatures —

Part 2: Mechanisms using a group public key

AMENDMENT 2

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

Partie 2: Mécanismes utilisant une clé publique de groupe

AMENDEMENT 2

Reference number
ISO/IEC 20008-2:2013/Amd. 2:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

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.

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

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Information technology — Security techniques — Anonymous digital signatures —

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

AMENDMENT 2

Clause 4

Add the following symbol:

$F(p)$ the finite field containing exactly p elements.

6.1

Replace the first sentence with the following:

> This clause specifies five digital signature mechanisms with linking capability.

Replace the text of NOTE 1 with the following:

> In the literature, the mechanism of 6.2 is called a list signature scheme, the mechanism of 6.6 is called a pre-DAA scheme and the mechanisms of 6.3, 6.4 and 6.5 are called DAA schemes. The mechanisms given in 6.2, 6.4, 6.5 and 6.6 are based on schemes originally specified in References [9], [6], [11] and [22] respectively, in which security proofs can also be found. The mechanism in 6.3 is based on a scheme in Reference [3] which is a minor modification of the scheme in Reference [4]; the associated security analysis is given in the full version of Reference [4].

6.6

Add new subclause 6.6 as follows:

6.6 Mechanism 8

6.6.1 Symbols

The following symbols apply in the specification of this mechanism.

— τ : a security parameter.

— P_1, Q_1, X_1, $Y_1, X_1^{'}, \tilde{X}_1, C_1, D, D', T_1$, $T_2, K_1, K_2, K, K_1^{'}, K_2^{'}, K', J, T_1^{'}, T_2^{'}, R, R', T, T', R'', T''$: elements of G_1.

— P_2, X_2, Y_2, $X_2^{'}, \tilde{X}_2$: elements of G_2.

— $x, y, z, x', z', c_k, s_x, s_z, c_k', s_1, u, v, w, v', r, s_2, k_r, k_x, k_z, c, z_r, z_x, z_z, c', s, l, k_s, c_m, \rho, c_m'$: integers in Z_p .

— n_I : an integer of size τ -bit.

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

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

6.6.2 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 τ as a security parameter.

b) Choose a bilinear group pair (G_1, G_2) of large prime order p , such that no efficiently computable homomorphism is known between G_1 and G_2, in either direction, and an associated pairing function $e: G_1 \times G_2 \rightarrow G_T$.

c) Choose two random independent generators P_1 and Q_1 of G_1 and provide additional information, denoted by π_{Gen} , that serve to demonstrate that these two generators were indeed chosen independently, that is without a potentially exploitable relationship between them (such as $Q_1 = [s]P_1$ for an integer s chosen by the group membership issuer). An example of how to verifiably select independent generators and to verify, using π_{Gen} , the correct generation of these generators, is given in Annex G.

d) Choose a random generator P_2 of G_2 .

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

f) Choose three random integers x , y and z in Z_p .

g) Compute $X_1 = [z]P_1 + [x]Q_1$, $Y_1 = [y]P_1$, $X_2 = [x]P_2$ and $Y_2 = [y]P_2$.

h) Choose two random integers x' and z' in Z_p .

i) Compute $X_1^{'} = [z']P_1 + [x']Q_1$ and $X_2^{'} = [x']P_2$.

j) Compute $c_k = H_2 (P_1 \,||\, Q_1 \,||\, P_2 \,||\, X_1 \,||\, Y_1 \,||\, X_2 \,||\, Y_2 \,||\, X_1^{'} \,||\, X_2^{'})$.

k) Compute $s_x = (x' + c_k \times x) \bmod p$ and $s_z = (z' + c_k \times z) \bmod p$.

l) Set $\pi_{\text{Val}} = (c_k , s_x , s_z)$ as a proof that the second component of the representation of X_1 in the base P_1 and Q_1 is equal to the discrete logarithm of X_2 in the base P_2 .

m) Output the following:

— group public parameter = (G_1 , G_2 , G_T , e , P_1 , Q_1 , P_2 , p , H_1 , H_2 , H_3),

— group public key = (X_1 , Y_1 , X_2 , Y_2 , π_{Gen} , π_{Val}),

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

NOTE 1 Examples of recommended parameters are provided in C.2.

Each entity involved in this anonymous signature mechanism should verify the validity of the group public key before using it. The group public key validity verification process includes the following steps:

a) Verify that P_1 and Q_1 were generated independently using π_{Gen} .

b) Verify the validity of the proof π_{Val} :

1) Compute $\tilde{X}_1 = [s_z]P_1 + [s_x]Q_1 - [c_k]X_1$ and $\tilde{X}_2 = [s_x]P_2 - [c_k]X_2$.

2) Compute $c_k^{'} = H_2\left(P_1 \| Q_1 \| P_2 \| X_1 \| Y_1 \| X_2 \| Y_2 \| \tilde{X}_1 \| \tilde{X}_2\right)$.

3) Verify that $c_k^{'} = c_k$.

c) Verify that $e(Y_1, P_2) = e(P_1, Y_2)$.

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

The group membership issuing process requires a secure and authentic channel between the group member 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 $n_I \in \{0,1\}^{\tau}$.

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

c) The member chooses a random integer s_1 from Z_p.

d) The member computes $C_1 = [s_1]Y_1$.

e) The member chooses a random integer u from Z_p .

f) The member computes $D = [u]Y_1$.

g) The member computes $v = H_2(P_1 || Q_1 || P_2 || X_1 || Y_1 || X_2 || Y_2 || C_1 || D || n_I)$.

h) The member computes $w = (u + v \times s_1) \bmod p$.

i) The member sends (C_1, v, w) to the group membership issuer.

j) The group membership issuer computes $D' = [w]Y_1 - [v]C_1$.

k) The group membership issuer computes $v' = H_2(P_1 || Q_1 || P_2 || X_1 || Y_1 || X_2 || Y_2 || C_1 || D' || n_I)$.

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

m) The group membership issuer selects five random integers r , s_2 , k_r , k_x and k_z from Z_p .

n) The group membership issuer computes $T_1 = [r]P_1$ and $T_2 = [x]T_1 + [r]C_1 + [r \times s_2]Y_1$.

o) The group membership issuer computes $K_1 = [k_r]P_1$, $K_2 = [k_x]T_1 + [k_r](C_1 + [s_2]Y_1)$ and $K = [k_z]P_1 + [k_x]Q_1$.

p) The group membership issuer computes $c = H_2(P_1 || Q_1 || P_2 || X_1 || Y_1 || X_2 || Y_2 || C_1 || s_2 || K_1 || K_2 || K)$.

q) The group membership issuer computes $z_r = (k_r + c \times r) \bmod p$, $z_x = (k_x + c \times x) \bmod p$ and $z_z = (k_z + c \times z) \bmod p$.

r) The group membership issuer sets (T_1, T_2) as the member's group membership credential and sends $(T_1, T_2, s_2, c, z_r, z_x, z_z)$ to the member.

s) The member computes $K_1^{'} = [z_r]P_1 - [c]T_1$, $K_2^{'} = [z_x]T_1 + [z_r](C_1 + [s_2]Y_1) - [c]T_2$ and $K' = [z_z]P_1 + [z_x]Q_1 - [c]X_1$.

t) The member computes $c' = H_2(P_1 || Q_1 || P_2 || X_1 || Y_1 || X_2 || Y_2 || C_1 || s_2 || K_1^{'} || K_2^{'} || K)$.

u) The member verifies $c = c'$. If the verification fails, the member aborts.

v) The member computes $s = (s_1 + s_2) \bmod p$.

w) The group member signature key for the member is (s, T_1, T_2).

NOTE 2 The group membership issuer can use the same value s_2 (for example $s_2 = 0 \bmod p$) for several executions of the group membership issuing process. In this case, the security of Mechanism 8 relies on the Pointcheval-Sanders (PS) assumption[24], instead of the q-MSDH assumption[25] if the group membership issuer uses a fresh random value s_2 for each new session of the group membership issuing process.

6.6.3 Signature process

On input of a group member signature key (s, T_1, T_2), a linking base bsn and a message $m \in \{0,1\}^*$ to be signed, the signature process takes the following steps. The linking base, denoted by bsn, is either a special symbol ⊥ or an arbitrary string used for the linking capability.

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

b) The signer selects two random integers l and k_s in Z_p.

c) The signer computes $T_1^{'} = [l]T_1$ and $T_2^{'} = [l]T_2$.

d) The signer computes $R = [s]T_1^{'}$ and $R' = [k_s]T_1^{'}$.

e) The signer computes $T = [s]J$ and $T' = [k_s]J$.

f) The signer computes $c_m = H_3(T_1^{'} || T_2^{'} || J || T || R || T' || R' || m)$.

g) The signer computes $\rho = (k_s + c_m \times s) \bmod p$.

h) The signer outputs the anonymous signature $\sigma = (T_1^{'}, T_2^{'}, J, R, T, c_m, \rho)$.

6.6.4 Verification process

On input of a message m, a linking base bsn, a signature $(T_1^{'}, T_2^{'}, J, R, T, c_m, \rho)$ and a group public key (X_1, Y_1, X_2, Y_2), the verification process takes the following steps:

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

b) Verify that $T_1^{'} \neq 0_E$.

c) If any of the above verifications fails, output 0 (invalid).

d) Compute $R'' = [\rho]T_1^{'} - [c_m]R$.

e) Compute $T'' = [\rho]J - [c_m]T$.

f) Compute $c_m^{'} = H_3(T_1^{'} || T_2^{'} || J || T || R || T'' || R'' || m)$.

g) Verify that $c_m^{'} = c_m$.

h) Verify that $e\,(T_1^{'},\ X_2) \times e\,(R,\ Y_2)\ =\ e(T_2^{'},\ P_2)$.

i) Optionally, call the revocation checking process.

j) If any of the above verifications (steps g and h) fails, output 0 (invalid). Otherwise, output 1 (valid).

6.6.5 Linking process

Given two valid signatures $\sigma\ =\ (T_1^{'},\ T_2^{'},\ J, R, T, c_m, \rho)$ and $\hat{\sigma}\ =\ (\hat{T}_1^{'},\ \hat{T}_2^{'},\ \hat{J}, \hat{R}, \hat{T}, \hat{c}_m, \hat{\rho})$, the linking process takes the following steps:

a) If $J\ =\ \hat{J}$ and $T\ =\ \hat{T}$, output 1 (linked), otherwise, output 0 (not linked).

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

6.6.6 Revocation process

Details of the revocation process in this mechanism are surveyed in Reference [10]. There are two types of revocation (private key revocation and verifier blacklist revocation) supported in this mechanism. Private key 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 $(s,\ T_1,\ T_2)$ is compromised, the group membership issuer puts s into a revocation list RL of this type.

— Given a valid signature $\sigma\ =\ (T_1^{'},\ T_2^{'},\ J, R, T, c_m, \rho)$ 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 $s' \in$ RL, verify $T\ \neq\ [s']J$. If any of these verifications 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. This revocation process allows to identify every group signature generated using this private key. If this key can be associated with a group member (e.g. by using contextual information), then no anonymity can be retained for this group member as their signatures can therefore be traced. This is a property inherent to DAA schemes. Thus, a careful assessment of the need for revocation and the consequences for the corresponding group member will be carried out before deployment.

Verifier blacklist revocation:

— If signatures were computed using a linking base bsn, a verifier can build its own revocation list RL corresponding to bsn. If the verifier wants to blacklist the signer of a valid signature $\sigma\ =\ (T_1^{'},\ T_2^{'},\ J, R, T, c_m, \rho)$, they put T into a revocation list RL of this type.

— Given a signature $\sigma\ =\ (T_1^{'},\ T_2^{'},\ J, R, T, c_m\ \rho)$ 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 $\hat{T}\ \in$ RL, verify $T\ \neq\ \hat{T}$. If any of these verifications fails, output 0 (revoked), otherwise, output 1 (valid).

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

7.1

Replace the first sentence with the following:

> This clause specifies three digital signature mechanisms with opening capability.

Replace the text of NOTE with the following:

> The mechanisms and associated security proofs in 7.2, 7.3 and 7.4 are based on References [17], [14] and [23] (an extended version of Reference [24]), respectively.

7.4

Add new subclause 7.4 as follows:

7.4 Mechanism 9

7.4.1 Symbols

The following symbols apply in the specification of this mechanism.

— P_1, S_i, T_1, T_2, K, K', T_1', T_2': elements of G_1.

— P_2, $X, Y, A, B, Y_i, C_1, C_2, C_3, C_4, K_1, K_2, K_3, K_4, K_1', K_2', K_3', K_4'$: elements of G_2.

— W, W', R, R_i: elements of G_T.

— $x, y, a, b, s_i, u, v, k_s, k_u, k_v, c, z_s, z_u, z_v, c', r, t, w, c_m, z, c_m'$: elements of Z_p.

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

7.4.2 Key generation process

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

a) Choose a bilinear group pair (G_1, G_2) of large prime order p, such that no efficiently computable homomorphism is known between G_1 and G_2, in either direction, and an associated pairing function $e: G_1 \times G_2 \to G_T$.

b) Choose a random generator P_1 of G_1 and a random generator P_2 of G_2.

c) Choose two random integers x and y in Z_p.

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

e) Choose a hash function $H: \{0,1\}^* \to Z_p$. Such a hash function shall be constructed as described in Annex B.

f) Output the following:

— group public parameters: (G_1, G_2, G_T, e, p, H, P_1, P_2),

— group public key: (X, Y),

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

The group membership opener key generation process takes the following steps:

a) Choose two random integers a and b in Z_p.

b) Compute $A = [a]P_2$ and $B = [b]P_2$.

c) Output the following:

- group membership opener public key (A, B),
- group membership opening key (a, b).

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 information associated with each registered user. This member-list **LIST** can be published but it will only be useful to the group membership opener.

The group membership issuing process is an interactive protocol running between the group membership issuer and a user U_i to create a group member signature key for the user. It consists of the following steps:

a) U_i selects six random integers s_i, u, v, k_s, k_u and k_v in Z_p.

b) U_i computes $S_i = [s_i]P_1$ and $Y_i = [s_i]Y$.

c) U_i computes $C_1 = [u]P_2$, $C_2 = Y_i + [u]A$, $C_3 = [v]P_2$, $C_4 = Y_i + [v]B$.

d) U_i computes $K = [k_s]P_1$, $K_1 = [k_u]P_2$, $K_2 = [k_s]Y + [k_u]A$, $K_3 = [k_v]P_2$, $K_4 = [k_s]Y + [k_v]B$.

e) U_i computes $c = \mathrm{H}(P_1 \| P_2 \| X \| Y \| A \| B \| S_i \| Y_i \| C_1 \| C_2 \| C_3 \| C_4 \| K \| K_1 \| K_2 \| K_3 \| K_4)$.

f) U_i computes $z_s = (k_s + c \times s_i) \bmod p$, $z_u = (k_u + c \times u) \bmod p$ and $z_v = (k_v + c \times v) \bmod p$.

g) U_i sends S_i, C_1, C_2, C_3, C_4 , c, z_s, z_u and z_v.

h) The group membership issuer computes $K' = [z_s]P_1 - [c]S_i$, $K_1^{'} = [z_u]P_2 - [c]C_1$, $K_2^{'} = [z_s]Y + [z_u]A - [c]C_2$, $K_3^{'} = [z_v]P_2 - [c]C_3$ and $K_4^{'} = [z_s]Y + [z_v]B - [c]C_4$.

i) The group membership issuer computes $c' = \mathrm{H}(P_1 \| P_2 \| X \| Y \| A \| B \| S_i \| Y_i \| C_1 \| C_2 \| C_3 \| C_4 \| K' \| K_1^{'} \| K_2^{'} \| K_3^{'} \| K_4^{'})$.

j) The group membership issuer checks if $c = c'$ and aborts if these two values are different.

k) The group membership issuer stores $(i, S_i, C_1, C_2, C_3, C_4, c, z_s, z_u, z_v)$ in LIST[i].

l) The group membership issuer selects a random integer r in Z_p.

m) The group membership issuer computes $T_1 = [r]P_1$ and $T_2 = [r \times x]P_1 + [r \times y]S_i$.

n) The group membership issuer sends T_1 and T_2 to the user U_i.

o) The signature key of the group member U_i is then (s_i, T_1, T_2).

7.4.3 Signature process

On input of a group public key (X , Y), a group member signature key (s_i, T_1, T_2) owned by the signer and a message $m \in \{0,1\}^*$ to be signed, the signature process takes the following steps.

a) The signer selects two random integers t and w in Z_p.

b) The signer computes $T_1' = [t]T_1$ and $T_2' = [t]T_2$.

c) The signer computes $W = e([w]T_1', Y)$.

d) The signer computes $c_m = H(T_1' \| T_2' \| W \| m)$.

e) The signer computes $z = (w + c_m \times s_i) \bmod p$.

f) The signer outputs the group signature $\sigma = (T_1', T_2', c_m, z)$.

7.4.4 Verification process

On input of a message $m \in \{0,1\}^*$, a group signature $\sigma = (T_1', T_2', c_m, z)$ and a group public key (X, Y), the verification process takes the following steps:

a) Verify that $T_1' \neq O_E$. If this verification fails, output 0 (invalid).

b) Compute $W' = e([z]T_1', Y) \times e\left([-c_m]T_2', P_2\right) \times e([c_m]T_1', X)$.

c) Compute $c_m' = H(T_1' \| T_2' \| W' \| m)$.

d) Verify that $c_m' = c_m$ holds.

e) If the above verification fails, output 0 (invalid), otherwise, output 1 (valid).

7.4.5 Opening process

Given a group signature $\sigma = (T_1', T_2', c_m, z)$, the member-list **LIST**=(LIST[1],...,LIST[n]) and a group opening key (a, b), the opening process takes the following steps:

a) Compute $R = e(T_2', P_2) \times e([-1]T_1', X)$.

b) For each $i \in [1, n]$,

   1) Recover (i, S_i, C_1, C_2, C_3, C_4, c, z_s, z_u, z_v) from LIST[i].

   2) Compute $Y_i = C_2 + [-a]C_1$.

   3) Verify if $e(T_1', Y_i) = R$.

   4) If the above equation holds, output i.

7.4.6 Revocation process

The revocation process is a membership credential revocation. The group membership opener revokes a user U_i by adding some element R_i (defined below) specific to this user in a revocation list RL.

Revocation can thus be global but also local as any verifier is able to manage its own revocation list by deciding whether or not to include R_i. In all cases, the revocation process has no impact whatsoever on the signature process.

— To revoke a user U_i, the group membership opener, taking as input the group opening key and **LIST**=(LIST[1],..., LIST[n]), proceeds as follows:

a) Recovers (i, S_i, C_1, C_2, C_3, C_4, c, z_s, z_u, z_v) from LIST[i].

b) Computes $R_i = C_2 + [-a]C_1$.

c) Adds R_i in RL.

— Given a group signature $\sigma = (T_1', T_2', c_m, z)$ and a revocation list RL, test for each element R_i in RL if $e(T_1', R_i) = e(T_2', P_2) \times e([-1]T_1', X)$. If this equality is satisfied by some element R_i in RL, then output 0 (revoked). Otherwise, output 1 (valid).

Annex A

Insert the following lines after `id-as-gpk-m-7 OID ::= { id-as-gpk mechanism7(7) }`:

```
id-as-gpk-m-8 OID ::= { id-as-gpk mechanism8(8) }
id-as-gpk-m-9 OID ::= { id-as-gpk mechanism9(9) }
```

Replace the line of `as-gpk-m-7` with the following:

```
as-gpk-m-7 |
as-gpk-m-8 |
as-gpk-m-9
```

Insert the following lines after `OID id-as-gpk-m-7 PARMS HashFunctions }`:

```
as-gpk-m-8 ALGORITHM ::= { OID id-as-gpk-m-8 PARMS HashFunctions }
as-gpk-m-9 ALGORITHM ::= { OID id-as-gpk-m-9 PARMS HashFunctions }
```

C.1.1, first paragraph

Replace the paragraph with the following:

> The following computational hardness assumptions underlie the security of the mechanisms specified in this document; 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], the static Diffie-Hellman (Static DH) assumption[8], the q-MSDH assumption[25] and the Pointcheval-Sanders (PS) assumption[24]. Table C.1 below summarizes which of these assumptions underlie the security of each of the mechanisms specified in this document.

Table C.1

Replace Table C.1 with the following table:

Table C.1 — Mathematical assumptions used in the mechanisms

	Strong RSA	DDH	SDH	LRSW	Static DH	q-MSDH	PS
Mechanism 1	√	√					
Mechanism 2	√	√			√		
Mechanism 3		√	√				
Mechanism 4		√		√	√		
Mechanism 5	√	√					
Mechanism 6		√	√				
Mechanism 7		√	√				
Mechanism 8		√				√	
Mechanism 9		√					√

C.1.7

Add new subclause C.1.7 as follows:

C.1.7 The q-MSDH assumption

Let (G_1, G_2) be a bilinear group pair of type 3 of large prime order p and an associated pairing function $e: G_1 \times G_2 \rightarrow G_T$. Let $P_1 \neq 0_E$ (respectively $P_2 \neq 0_E$) be given in G_1 (respectively in G_2). Given q pairs $([x^i]P_1, [x^i]P_2)$, for i from 1 to q, and a triplet $([a]P_1, [a]P_2, [a \times x]P_2)$ for some integers a and x in Z_p, the q-MSDH assumption states that it is computationally infeasible to generate a tuple $(w, R, [x+w]Q, [a/R(x)]Q)$, where Q is an element of G_1 different from the neutral element of this group, R is a polynomial of degree at most q and w is a scalar such that $(X + w)$ and $R(X)$ are relatively prime.

C.1.8

Add new subclause C.1.8 as follows:

C.1.8 The Pointcheval-Sanders (PS) assumption

Let (G_1, G_2) be a bilinear group pair of large prime order p and an associated pairing function $e: G_1 \times G_2 \rightarrow G_T$. Let $(P_1, [y]P_1)$ be given in G_1 and $(P_2, [x]P_2, [y]P_2)$ be given in G_2 for some integers x and y in Z_p. Assume that an oracle can be called that answers queries s in Z_p by a pair $(Q, [x + y \times s]Q)$, where Q is a random group element of G_1. Let this oracle be called with the following queries $s_1, s_2, ..., s_m$. The PS assumption states that it is computationally infeasible to generate a triplet $(t, R, [x + y \times t]R)$, where $t \notin \{s_1, s_2, ..., s_m\}$ and R is an element of G_1 different from the neutral element of this group.

C.2, first paragraph

Replace the paragraph with the following:

Mechanisms 3, 4, 6, 7, 8, 9 all make use of a pairing function. Methods of generating pairing-friendly elliptic curves are given in ISO/IEC 15946-5. For current standard security levels (128-bit, 192-bit and 256-bit), the curves parameters proposed in section 7 of Reference [21] are recommended.

C.2, third paragraph

Replace the paragraph with the following:

The following security parameters are recommended:

- **Mechanism 1:** For 112-bit security strength, the following parameters are recommended: l_p (1 024-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 (2 048-bit), l_f (104-bit), l_e (368-bit), $l_e^{'}$ (120-bit), l_v (2 536-bit), $l_\varnothing$ (80-bit), l_H (160-bit), l_r (80-bit), l_s (1 024-bit), l_Γ (1 632-bit), l_ρ (208-bit).
  - For 112-bit security strength, the following parameters are recommended: l_n (2 048-bit), l_f (112-bit), l_e (544-bit), $l_e^{'}$ (128-bit), l_v (2 720-bit), $l_\varnothing$ (128-bit), l_H (256-bit), l_r (128-bit), l_s (1 024-bit), l_Γ (2 048-bit), l_ρ (224-bit).
- **Mechanism 5:**
  - For 80-bit security strength, the following parameters are recommended: K_n (1 024-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 (2 048-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 (3 076-bit), K (256-bit), K_c (256-bit), K_s (128-bit), K_e (832-bit), $K_{e'}$ (60-bit).

Annex D, eleventh paragraph

Replace the paragraph with the following:

There are five revocation mechanisms specified in this document. Credential update is a type of membership credential revocation, where each signer updates its credential so the proof that the membership credential of the signer is not in the list in inherited in signature generation.

Table D.1 summarizes which mechanisms are global revocation and which are local.

Table D.1

Replace Table D.1 with the following table:

Table D.1 — Categorization of revocation mechanisms

	Private key revocation	Verifier blacklist revocation	Signature revocation	Membership credential revocation	Credential update
Global revocation	√		√	√	√
Local revocation	√	√	√	√	

Table D.2

Replace Table D.2 with the following table:

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

	Private key revocation	Verifier blacklist revocation	Signature revocation	Membership credential revocation	Credential update
Mechanism 1	√	√			
Mechanism 2	√	√			
Mechanism 3	√	√	√		
Mechanism 4	√	√	√		√
Mechanism 5					√
Mechanism 6					√
Mechanism 7					√
Mechanism 8	√	√			
Mechanism 9				√	

Annex D, NOTE 2

Replace the first three sentences of NOTE 2 with the following:

> In mechanisms 1-4 and 8, it can be possible for the holder of a revoked private key to be "framed" for signatures they did not create. If a malicious entity learns a group member private key from the revocation list, it can 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 mechanisms 4 and 8, the malicious party can obtain a valid group membership credential by re-enrolling with this private key.

Annex D, NOTE 3

Replace the first sentence of NOTE 3 with the following:

> In mechanisms 1-4 and 8, the group membership issuer can create membership credentials on a group membership private key that it learned.

E.8

Add new subclause E.8 as follows:

E.8 Mechanism 8

Security parameter

$\tau = 128$

The groups G_1, G_2 and G_T shall be constructed by using the BLS-462 curve defined in ISO/IEC 15946-5:2022, D.3.3.

This curve is defined by the parameters:

$u = -2^{77} + 2^{50} + 2^{33}$

p =

```
1555 5545554D 5A555A55 D6941493 5FBD6F1E 32D8BACC A47B1484 8B42A8DF FA5C1CC0
0F26AA91 557F0040 00200005 55554AAA AAAC0000 AAAAAAAB
```

$a = 0$

$b = 4$

(uncompressed) G =

```
023EEF43 38128200 BF5BF4FE 4BB7934B 9DFB4DB5 B8D3590C 01362DB4 040672C0 8172E8CF 3795B85F
1D89DDBF CC047A20 E4D33AAE 107E127F 4EC2039E CE0C0947 FEB77E57 8B058D1D 4D57E0A4 769D50A0
22FC74EF D181D31F A66BDFCE 38A80BDA B1B73B90 E59CFD7B 1402BC10 B4B912C3 F433F34A
```

n =

```
FFFFF F7FFFC01 80017FE0 5FD000E8 01FC017F FC800011 00007FEF FFEFFFFC 00000000 00000001
```

(cofactor) r =

```
1555554 FFFFD55A AAB01556 AAA7FFFE AAAAAAAB
```

In the following numeric example, P in G_1 is represented as $P.x$ || $P.y$, where $P.x$ and $P.y$ are elements in $F(p)$. In the same way P_2 in G_2 is represented as $P_2.x$ || $P_2.y$, where $P_2.x$ and $P_2.y$ are elements in $F(p^2)$.

SHA-256 is used as the underlying hash function.

Group membership issuer key generation

x =

```
00049733 F8F718B4 E10E5561 C41F228B B95FDB87 6341998A 4565895C 7B3FF872 716ECEAD 22B12685
```

y =

```
000DD4A8 D8D28000 79FB33B7 A8B44CF4 3C897972 8C6D631A B72DFAED B1C2D991 C9871849 DCA89CE0
```

z =

```
00073F1B C112FA0B 976C7BF5 56B09829 C4C26810 ED139AAF 4F44967A 80AB930E F96F1AAC 14C5B248
```

P_1 =

```
0D1D6167 8F1BD8CE EEC6C5CF 6B10B11C 7074E954 F64C3C37 9F1BD278 992F0827 3FE13089 A115260C
C033E469 DC88C8A9 EF68B1C7 B4B2CF4E 2ACA10F4 3C6BFA15 4EB071ED 4ECDB73C 90BD8FE1 1ACB9693
EC020AAC 772FEB72 B57039BF DA6D7F45 97C65249 B7CB371C 959EB84B 9939BE5B 16FB3541
```

Q_1 =

```
0CAC1679 151DBBFE 05DCFCFC 098D0845 A49E625D 9FD1F057 1BE9DCF7 5F722823 5888646E 56D4D26D
E47C8EFB FE080B93 4CA87537 89C8D13E 16AF10EE 062BC15C 6D8A821A A2EEE5D3 12A1D64D D3350103
2EBC625D 00D975B1 ED43762F E98DF984 0863E731 A9CB7C25 FE3E199E 0814935F FD249D4F
```

P_2 =

```
0AA6EE37 803835BC 41CB01B5 27BE2C3D A3FEC9D7 3CAA9147 D67E5BBE 7776E1BB 77A15BC0 4EA31410
6B13FD12 8C017B49 A86E5CA4 06F638C6 B25E09F7 6927330E B7AFB96F D63DADEF 95E66AE5 75656DD4
CB08CC46 AD80CD1C 041FA96A 9A0F8519 46745EDC 44BABBC6 A8EB06A2 63AE805A 741F43A8 00F38198
DE2EFE97 FD6C0A02 EFFF5C11 FEA60504 697E18A0 D6C35073 69B167F0 58F29647 77309E79 211FF700
67D6C576 32353791 7BAB03C5 07FD0FC7 FA314144 8DFC13F5 4B7ADDCA 51FC4A47 45FE427E E509D485
A64E8BC9 116F5D83 70F237CF 063B8446 BF287E4D 2539BF44 EA4B8C12 965786C1
```

X_1 =

```
029C24BA 66D921C0 0833BF32 48C0D070 4F63194E 424E4482 D9822695 B8AD6E62 7F5F75E2 21D555B6
7D61E315 14C6706E CD988A9F 3246644A 248B13BB 618C9D81 3B6A9644 E1169A7F D00EC234 CD03A374
01F67112 3E65A16F 99FB1915 027F2031 F3FE09E0 E5962BE8 F4A05F28 83706988 4FF5449A
```

Y_1 =

```
0BBA851D AF4ECEF2 25B27CC6 EF1C5B80 E7A77329 94B7C216 922963DD 75816C6E 2B6762CA 0AEE7D23
08952478 F881B1DA 81B09F63 6F32A9DF AEA506CA D30DE5E3 4542C843 52DCA8A9 EA1884AD A5685BDF
8EDF8A2F B012A7A2 2F07C4C8 7265DDDB A854174F 83BFA29E C519C9FC 3002A4A6 0050A7D6
```

X_2 =

```
052B4CF9 16462ACE ADAB3220 1556CA3F D11CA47B 8B55086F 25AE0F7D 87E149BE 4EE9BFF2 2EB2A9A1
30E06BF2 19448E3E 5EA3855C CB227FEE F3C81432 7C348579 33CF4A5D 764539E4 8DE74D6A 639FF75B
10DB90C4 700B535C F7B8626E 06077DC6 0EEC2B00 A3E61DF2 19DEE937 F062038C 198A34D3 0870C7B4
BFC06E11 D814F7D5 58A31925 BD51EF5C 04CEBB76 E97A60E2 598653EA 7FA9B5DC 1A50E2A9 FABB33C5
C999F1F0 6D07AE93 A7EAA496 21460D87 3DF3F046 9129BF15 F3344BED B6B817E5 27A4DEAD 48A0DC74
B1D59CC5 E781B755 7CC8ECEA BBE29AC2 8D85CD3C 3D97B592 F428C975 24E051CA
```

Y_2 =

```
0E3E4BFF F58E6F0D 99D305AF C023232A 7B290855 DA20DEBE 78D04B30 FFBF5A51 6E5AE687 04C45121
920A9919 823D5711 E5F318E4 D92FDED4 3FF10DC6 3707BABD F94F9F6C B5E7ECD5 A8B8E27A 7FAE93FE
200D172B 23148B57 220E9A99 C1696F08 50FBD981 093B630B D6D6549F 50122B8C BD99715F 0EA1B461
B93A5887 F1DB2D37 D1464B78 5313EBA8 D83660DC 86A99BF5 8A7A8632 B95B28CF 06306EAA 2C884A06
2A590C6E 254C96C6 FA0454DF 2A1E0F29 B393319F 0E0F5241 B86E65E5 58E5CC63 7EC13DC3 1F355F67
45F7811E 8026DE77 FCFD967C D9C6DB73 F1408300 6F5CF1D5 86A35E96 DC790765
```

Group membership issuing process

n_I =

```
445F8AEC 300DCF97 88FC08A7 6E7A2C38
```

s_1 =

```
0003944C B4EFC1BC 1B9345C1 ECFFD432 ED9F99B9 54B8AEAD 3700D233 6F329249 3CB91195 6B4BA2B4
```

u =

```
000539E3 E744AA6E 8F364E86 6BB5C9E1 625E6B5A 11043986 264A9219 0ACB5BB5 8E743307 2127727A
```

C_1 =

```
133B1A42 DCFEC9B9 6931B63D 1CEA686B D04B6E08 975EF31D 04DFDC8D 1CEA5483 EB861BFC E4FE5DFD
DD44B094 BB50825F 8373910E 29D0F36B 849514F3 45DE9AA9 213EDA5F 6C054EEF C409C497 EA355639
1039E6E1 E701283B 8D155763 49A3AFA3 DCFD7F3B 797D9D6A B33F6D53 E6B618E8 884BC737
```

D =

```
003B115D 9906A915 31B10D73 920821D3 F12DDC38 9FD8DCB2 29FC33E0 B14F0DE5 61DE19B0 4F9445BD
61EF30EA 530C22CB 73089FBD DE1BEA11 895D14F2 1FECCCDD D45DA69F C16706D9 B11EE000 A00BF518
E0A462A5 34866595 BD7B1091 0EFECA6D 861BA676 5C9FA417 6A031F1F 4B1949DF CA4A46E9
```

v =

```
AD00E959 A1ABA12E F887558D 04D1B667 529ECE62 3726221A 38C2116C F0118D3E
```

w =

```
000ED654 F6672247 AD47BFF2 57F67A95 0CF1B544 9F71EA0E 4F3182D8 22367784 975100B4 4A5D579E
```

D' =

```
003B115D 9906A915 31B10D73 920821D3 F12DDC38 9FD8DCB2 29FC33E0 B14F0DE5 61DE19B0 4F9445BD
61EF30EA 530C22CB 73089FBD DE1BEA11 895D14F2 1FECCCDD D45DA69F C16706D9 B11EE000 A00BF518
E0A462A5 34866595 BD7B1091 0EFECA6D 861BA676 5C9FA417 6A031F1F 4B1949DF CA4A46E9
```

v' =

```
AD00E959 A1ABA12E F887558D 04D1B667 529ECE62 3726221A 38C2116C F0118D3E
```

r =

```
000684F8 CA32C059 5B81CF59 4D5EDF59 55E63C05 A457D624 9AA3F0D6 BBBFA775 F314FFC9 6CFFAE84
```

s_2 =

```
000735CF BA8ACE92 005CD792 2643BA25 50B2D7E4 49411F11 1F97B6C8 803BBF11 6E6C0450 99EC2767
```

k_r =

```
000B7DDF D567AA6F 83081001 A230592D 706078A7 37D04810 24F6E542 323ED467 460337AA FFAE5F38
```

k_x =

```
000BC74A 86C37A14 8E4434B0 A4C254BA F0A28DFB 95715AD9 6B534FE5 35297DC2 E1E4A40E 84DF6D9E
```

k_z =

```
000B566B 9ED799A6 BB8061AF C0338435 0D0377A8 F3C75E9A 2BE1631E B45D0FAC 25D6D986 8525F779
```

T_1 =

```
06F23C80 5437A50B B83D974E 658BD70F B7063418 54E1E503 D7E33197 AB2A22F6 210FB1DA 12F834E3
B1F52931 22666202 B1C6CA4E B9EA14A4 0E8913B9 1D94B361 862A653A D4A71EA0 26530FE3 5FE9F688
FA1AE672 9FCFE016 51DAC03A BE0C2C5B 312508AA C44EE867 BF3C8087 88A8FFD9 A57BE4FE
```

T_2 =

```
0991CE1C 6575C5CE FFBC6967 B2AEAD10 A951D0D1 B56E9DCE 54D37BF3 EFC1F217 C86A0204 53FC4CD2
1E721469 0C3B7B98 9785E03F B20FB7EE 3EC1054D D8E9B72A C40C5F18 BE6EDF61 38DF60F8 3A7E991C
167ADD0D 4789C399 0327F749 471C431C D8A8905E B9029BB1 F3E9500C 49E143CB 46537ECE
```

K_1 =

```
0235E50D 6D5B8298 5997ED82 F142ED0A AC30E35C 3DD88689 70033588 E3285718 D1265C91 F0D72960
6647AE1E 1034898F E437E0F1 0466A09D 23BE0471 BE01C0CB C5F79EDC 613733B3 2AEF2A66 259C759B
435AE85A ACC7DC56 73528770 5F65697B 9424E454 B655B617 0D151723 F4665C96 683104FE
```

K_2 =

```
10001B14 B4474D8A 5E4D05BC 7B795DA7 E28C5DEE 6C9D6BE8 EDD152E7 A870B7A7 56EAC839 7FCCDD56
E26342E5 C25D5AE2 6A044413 4334142F 4B641407 3ACF0276 E8D74241 9C4CA853 2A9F424E 382750CF
179CC3AF E5B3F054 61A27C74 56DA47B3 3FDEDAA1 8B296290 4A1A0CC9 4775BC93 E166D3F3
```

K =

```
0891B7E8 91F1FE93 7373AE84 1E78C591 42BC0796 5F0361F5 4E5ACF39 B2922EFC 9F843D50 38FA13CC
C24F887D 5FD824B7 8B89B4AB 7C1C5388 1BE513BC 1C946C94 BFAD98B5 5B960D95 32382813 7610A288
61F98CC0 0E4443D6 2DD27A49 7D87E992 9DBDAF4B E126FEEC 6D6F66C8 CDF31D2C 8C7FC9AD
```

c =

```
6BBA6AC3 D099E294 FF6A0A16 6F4E470C B9F3C96C 381279D5 5123A7DE 22FB2A1B
```

z_r =

```
0005D015 C7FAA5DA 8A331D22 04E783CB 412444EA C7CC25F7 ECFFB8EF 1C45264D DCF7EDDD 4A138B66
```

z_x =

```
0008DB60 D4B86A68 F88C66A1 1F27A927 9AFE9E67 762583A6 3BCDCB06 FFC8EAEF 351E76FF DA80706B
```

z_z =

```
000A2A2C 9F20F63A 6AC48E76 41E064F8 E4EF46A8 1327BE3D 364ACCEE 5F85DE11 0C993886 64D691E5
```

K'_1 =

```
0235E50D 6D5B8298 5997ED82 F142ED0A AC30E35C 3DD88689 70033588 E3285718 D1265C91 F0D72960
6647AE1E 1034898F E437E0F1 0466A09D 23BE0471 BE01C0CB C5F79EDC 613733B3 2AEF2A66 259C759B
435AE85A ACC7DC56 73528770 5F65697B 9424E454 B655B617 0D151723 F4665C96 683104FE
```

K'_2 =

```
10001B14 B4474D8A 5E4D05BC 7B795DA7 E28C5DEE 6C9D6BE8 EDD152E7 A870B7A7 56EAC839 7FCCDD56
E26342E5 C25D5AE2 6A044413 4334142F 4B641407 3ACF0276 E8D74241 9C4CA853 2A9F424E 382750CF
179CC3AF E5B3F054 61A27C74 56DA47B3 3FDEDAA1 8B296290 4A1A0CC9 4775BC93 E166D3F3
```

K' =

```
0891B7E8 91F1FE93 7373AE84 1E78C591 42BC0796 5F0361F5 4E5ACF39 B2922EFC 9F843D50 38FA13CC
C24F887D 5FD824B7 8B89B4AB 7C1C5388 1BE513BC 1C946C94 BFAD98B5 5B960D95 32382813 7610A288
61F98CC0 0E4443D6 2DD27A49 7D87E992 9DBDAF4B E126FEEC 6D6F66C8 CDF31D2C 8C7FC9AD
```

c' =

```
6BBA6AC3 D099E294 FF6A0A16 6F4E470C B9F3C96C 381279D5 5123A7DE 22FB2A1B
```

s =

```
000ACA1C 6F7A904E 1BF01D54 13438E58 3E52719D 9DF9CDBE 569888FB EF6E515A AB2515E6 0537CA1B
```

Signature process

message m =

"Data to sign"

J =

```
0C2C0E28 D59346D8 45ED29E5 CFE7CBBF E18CD573 6966825D 94CC1140 1A529B0F 8CB37561 D79EDCC2
680034F3 4724A0FF 73A3A1E9 759452E5 ADFD0D73 724493A8 B6A1B221 4E3DF3F3 F1B7225F 1798C92B
8B6849D2 9C671AAD 6873CEC4 84D81E64 42AB0FCE 1601690D 7020B6C0 EEB301BB EB66D692
```

l =

```
0002C46A C93BB89C 505E5723 4E320D2C 0437AA72 2647761F C3941FC0 7252CA6E 7F96FFFA 48CCB186
```

k_s =

```
0008A1E1 F3D62F50 DF798B66 063511B3 6F8EE4FA 9F0CD5AB 668FC567 4E3D1028 9D330932 6E17A867
```

T'_1 =

```
0AAE5075 82D3A1C7 49A60007 6F5499B5 99749898 25588C4A DF607DBE 5BD7F575 6B61D1E7 E633B39F
1F768CA4 1423A7C6 23FF6A0D 1A838D99 100E0891 C2E98E2E 58AB4369 1E7FC7E4 4F0D4614 D254C9C3
78AE65FA BD1A79F5 D9BAAAE0 2FEDF765 9C69347B 90DF8D45 7CB0C42D EB1700D9 2A79D1F8
```

T'_2 =

```
0E244CAE F717ACCA A842CF6E 46F47291 C8CC9692 DAF73BF1 8E99AF16 5EA4746B 5E1FAEC8 B0A4CACB
50140398 F9C7358F 1AF9C112 C27FC699 48470FF0 6FB24ACC C9F9F347 595C33DC E17EC565 612DBB6A
E005AF0B E6A30084 0C87F1D0 6B5B356B E658BCF1 E1A7E3D4 0E735374 3AD35D1E EB9D3808
```

R =

```
04DCF8C3 07861012 A4F9CEFF D3FEECFD 6653FCC8 0A30253F 693C0613 DEC29AF3 2DA63E88 2B94BCF8
30B7CD9B 328C4FD0 75A70BDC 3BB2111F 6F021459 77EE545C 0ECB280B 613BE3D4 44036EC0 7DB5EB42
96713BAB 30F66A20 99E9DB76 8F5AA7E4 A1668921 AF1BBFF6 AC82D917 1E35E638 6A50F08B
```

R' =

```
03759853 71F19A13 B0A64FD8 3960506A 69DDCD58 D6C7B891 45BBECA7 834C7802 94AF5738 58D2946B
0B67D554 76592CF3 3060AFA7 A5F5FC0D 781901DD 6F5D6C24 DC18B580 3738A892 BB9BF2E3 44C01250
18EA0789 ADA51A93 F52D1125 6D677647 6DE9BF20 FE0774CA 82B5B5DC C34EEE82 A2D7FF84
```

T =

```
0D4CCC68 B34CA281 03A9277F B8BC1CE2 C5118834 BD91F432 913E8847 E6284255 383ABBCA 8ADE57A0
C26FFB07 C1C672BA 49A433F3 992B2CB2 DB960E2D 28FC7CAE 16AD6E51 1B9DE20D BE9032B0 A7784006
CF9DA520 99123CBB 324978A5 1A19C6BB 218A2AC9 E8BE3617 69110A36 A804BCC1 A92195C3
```

T' =

```
1216EFBC 335F37F0 6BFB3EA3 F05FDDB2 C2BD165C FD7F700A 9AA47985 92B1243F 7E790E18 FBF69AC8
B7393AD0 C31E8E75 A775BFF2 0D58EA1F ADF812F9 B9681F58 008ECD9A E987566A 55200402 58EACAFB
53DBC3E2 9CB75E79 BEE6288B FBE3D7BC C8350455 6407D3DE FBFB7D0A E3BDC93F 107FB4E9
```

c_m =

```
7F9BA86D 285BB773 E4085B8A 23DEA69C B1670442 02520155 72A45688 2F611E9E
```

ρ =

```
0001BBEF 872780BD D763A3B1 A1B5BCEC 090D907D 811BB727 771C9DA0 B1216A2D 60A45167 667A46C8
```

Verification process

R'' =

```
03759853 71F19A13 B0A64FD8 3960506A 69DDCD58 D6C7B891 45BBECA7 834C7802 94AF5738 58D2946B
0B67D554 76592CF3 3060AFA7 A5F5FC0D 781901DD 6F5D6C24 DC18B580 3738A892 BB9BF2E3 44C01250
18EA0789 ADA51A93 F52D1125 6D677647 6DE9BF20 FE0774CA 82B5B5DC C34EEE82 A2D7FF84
```

T'' =

```
1216EFBC 335F37F0 6BFB3EA3 F05FDDB2 C2BD165C FD7F700A 9AA47985 92B1243F 7E790E18 FBF69AC8
B7393AD0 C31E8E75 A775BFF2 0D58EA1F ADF812F9 B9681F58 008ECD9A E987566A 55200402 58EACAFB
53DBC3E2 9CB75E79 BEE6288B FBE3D7BC C8350455 6407D3DE FBFB7D0A E3BDC93F 107FB4E9
```

c_m' =

```
7F9BA86D 285BB773 E4085B8A 23DEA69C B1670442 02520155 72A45688 2F611E9E
```

E.9

Add new subclause E.9 as follows:

E.9 Mechanism 9

The groups G_1 , G_2 and G_T are constructed by using the BLS-462 curve defined ISO/IEC 15946-5:2022, D.3.3.

This curve is defined by the parameters:

u = $- 2^{77} + 2^{50} + 2^{33}$

p =

```
1555 5545554D 5A555A55 D6941493 5FBD6F1E 32D8BACC A47B1484 8B42A8DF FA5C1CC0
0F26AA91 557F0040 00200005 55554AAA AAAC0000 AAAAAAAB
```

a = 0

b = 4

(uncompressed) G =

```
023EEF43 38128200 BF5BF4FE 4BB7934B 9DFB4DB5 B8D3590C 01362DB4 040672C0 8172E8CF 3795B85F
1D89DDBF CC047A20 E4D33AAE 107E127F 4EC2039E CE0C0947 FEB77E57 8B058D1D 4D57E0A4 769D50A0
```

```
22FC74EF D181D31F A66BDFCE 38A80BDA B1B73B90 E59CFD7B 1402BC10 B4B912C3 F433F34A
```

n =

```
FFFFF F7FFFC01 80017FE0 5FD000E8 01FC017F FC800011 00007FEF FFEFFFFC 00000000 00000001
```

(cofactor) r =

```
1555554 FFFFD55A AAB01556 AAA7FFFE AAAAAAAB
```

In the following numeric example, P in G_1 is represented as P.x || P.y, where P.x and P.y are elements in $F(p)$. In the same way P_2 in G_2 is represented as P_2.x || P_2.y, where P_2.x and P_2.y are elements in $F(p^2)$.

SHA-256 is used as the underlying hash function.

Key generation process (Group membership issuer key generation)

x =

```
000B5604 DC5A2ACE BB955263 0E274523 C8BF23D3 361862B6 505E5349 CB572B4C 82E5D517 360B4658
```

y =

```
000C56B6 865B86F7 F8FA614E 12254EDF 05E04C55 70E3984A E7E77495 09DF654B BAF72CCD 19F03064
```

P_1 =

```
023EEF43 38128200 BF5BF4FE 4BB7934B 9DFB4DB5 B8D3590C 01362DB4 040672C0 8172E8CF 3795B85F
1D89DDBF CC047A20 E4D33AAE 107E127F 4EC2039E CE0C0947 FEB77E57 8B058D1D 4D57E0A4 769D50A0
22FC74EF D181D31F A66BDFCE 38A80BDA B1B73B90 E59CFD7B 1402BC10 B4B912C3 F433F34A
```

P_2 =

```
05D75191 145C880D 428796E8 C5F45F4E 0DBCFA32 F8EC80BB BD0B52B2 DAFFA29D 0CA2AEFF F23A4E9D
8E2C7B83 D1AB0935 1EFFA7AB 256BE294 2EE813E8 FF40EEA4 537DC516 11128F1E A2A28DFF E1C5FA59
C36F9004 0069E915 1272E89E 3B565460 0328F730 BDF1495C CEE1220E B5CFD3A3 658C672C 15094CE2
0B17CC08 B8E91CA3 3A298EA1 6A77C8BB 9BC00C1B 31AA0C44 E45AEFCA BE8387AE DACD824B 62EA6887
2BC3F1B8 B87EC1F0 94FCC273 114B05D0 ED46F901 EC8A15A7 0215A989 BA2F7998 50659059 6008D58A
BC9894A4 73DC5527 08327B63 99A0C680 772D1308 DA374C69 C29403ED D4573F1E
```

X =

```
1257D198 72120B12 CED3E9E4 B0413765 E200983D 0F53C255 374E0A3B D66934C2 B2321523 D6D83864
904FE52A 1689315E 73B61B19 A13B4374 4E7911FC 7700B52C CE6B234E BA3C0E8D 71AF4761 DAE9B3E1
3198158D 10A704BB 9EDF233A D81CB64A 842E5D58 BD8776A4 96DD4EE1 36790CEF D4E8D843 0512B1CD
6EC06133 B2147E78 7158BF42 BEB79FF8 476EACE3 835EA421 FC14D870 542AE488 F31529B8 8553B892
1BC8DFEA 4C023166 47790C2D EBF009E0 E2011D86 7B791DDB B8998C5D B9CC4F7B 2B13F7E8 7CB94FB6
047FD28A 554311EF 3B0E3282 6F1BF222 A6B43960 A22544FE 75B76C61 3C9CC805
```

Y =

```
07CFA2EE D283DB5A 49B539F2 235D8D1E B180851D 560523AB 1296EBB0 D3289452 104EF9E8 54E6833A
FA8232C1 149B267F D8AD11B8 F8F2C1FD 96E51481 7A2C24C5 FDD3F6C7 7A6EED9A B11EF02A FD73FD5E
0BDFF105 261D7EDA 9F98C3A5 78904879 78B9B97C F62704F1 AF549582 DBAD11E0 0773FA4B 07189024
1CC8FA45 7E20F037 83A6460D 85D19791 FB5A4AB1 8CC18FE8 BB95722D F8C252BB 240DAD9E 8787BE99
7EB52600 3349BEC7 70A5D92C D38D151E 962537FD 1D41028C 2D39EDBD 8F18CF49 DE2B8784 A3F2ECB7
370816E8 E7736042 BB001583 17FBC4C4 9DCB76F0 E2C19989 7ECD6926 152FFD6B
```

Key generation process (Group membership opener key generation)

a =

```
000ACC46 870B64A2 BC56324D 5C635932 B1896AE3 CEE17C28 616A6121 28FD7CF3 842B5A02 455AA300
```

b =

```
0005E187 142B75EF 99F63979 D46ABC5D 56C1D1E1 31D100AB F017449E CAB8EF70 764B094F 4C10C421
```

A =

```
0218EEE7 0937D1AE 9AA81CAF AF5C6891 09AE6171 F33695C6 2902D100 667B87E9 813C1679 652650CC
32BF6968 96088DA6 9B3BBCE2 2C027596 14BF0C23 A027DC75 6ACD8817 C54B6C27 C1D61B45 CAEE5FBE
C3574881 8BAC1F18 AFC3C20D 6574F856 06A036B1 38457EA2 2D7C9ADE 9E2CAE70 3F2B1F66 147E4B9D
30CD75BB 6CE5737A 6A56B0D2 29283DEA 9336DF2F C5A1EED3 314B8EE2 349C0491 BD51569D 7A793529
02A81669 11676EC8 BD197F12 5F200E1B 7AE79BE3 7BB2C5BA AD6F213B 1D72C4FD 07F5048D 89C04737
9B9AF5AF 59FDB062 D6E905D4 F9BC3D34 AD2E3408 C64D8C60 E7EC8630 7E5405EE
```

B =

```
01FD0DF6 C9214E9B A6A650A2 407FF8A9 FFE9C720 E66FA61B 529B2415 37227F23 9E0518E7 C5876DE2
7046938B 4524F66C 2E904A13 50DB4E0C B1AD048C 3FCC2EF3 3F5DE7D5 41721930 C592F1E4 D4F01B3B
A1A28C3E 1EEFA0DC 93818E0B 8586AA54 1BEBF53E 432FCAB2 60C5785E 74DBBE15 D4FED0B5 009087B3
49C464F6 82D3F213 C08EACE9 B74FC508 B019596A 34C79A39 7A261413 7193FA5E 175FD842 F39D5C1B
EC70E932 3D3FDE2C 735907BA A5AE0DBA 439FB7F3 9443F7DA 3DD9E149 E8AFAD00 0C95C95B 054EFA7C
F400D27A 13B8794E 00C04118 66D0E5EB 1F94E3F9 5D11EE51 FCA02E1A 1E2F7F66
```

Group membership opener opening key

a =

```
000ACC46 870B64A2 BC56324D 5C635932 B1896AE3 CEE17C28 616A6121 28FD7CF3 842B5A02 455AA300
```

b =

```
0005E187 142B75EF 99F63979 D46ABC5D 56C1D1E1 31D100AB F017449E CAB8EF70 764B094F 4C10C421
```

Group membership opener public key

A =

```
0218EEE7 0937D1AE 9AA81CAF AF5C6891 09AE6171 F33695C6 2902D100 667B87E9 813C1679 652650CC
32BF6968 96088DA6 9B3BBCE2 2C027596 14BF0C23 A027DC75 6ACD8817 C54B6C27 C1D61B45 CAEE5FBE
C3574881 8BAC1F18 AFC3C20D 6574F856 06A036B1 38457EA2 2D7C9ADE 9E2CAE70 3F2B1F66 147E4B9D
30CD75BB 6CE5737A 6A56B0D2 29283DEA 9336DF2F C5A1EED3 314B8EE2 349C0491 BD51569D 7A793529
02A81669 11676EC8 BD197F12 5F200E1B 7AE79BE3 7BB2C5BA AD6F213B 1D72C4FD 07F5048D 89C04737
9B9AF5AF 59FDB062 D6E905D4 F9BC3D34 AD2E3408 C64D8C60 E7EC8630 7E5405EE
```

B =

```
01FD0DF6 C9214E9B A6A650A2 407FF8A9 FFE9C720 E66FA61B 529B2415 37227F23 9E0518E7 C5876DE2
7046938B 4524F66C 2E904A13 50DB4E0C B1AD048C 3FCC2EF3 3F5DE7D5 41721930 C592F1E4 D4F01B3B
A1A28C3E 1EEFA0DC 93818E0B 8586AA54 1BEBF53E 432FCAB2 60C5785E 74DBBE15 D4FED0B5 009087B3
49C464F6 82D3F213 C08EACE9 B74FC508 B019596A 34C79A39 7A261413 7193FA5E 175FD842 F39D5C1B
EC70E932 3D3FDE2C 735907BA A5AE0DBA 439FB7F3 9443F7DA 3DD9E149 E8AFAD00 0C95C95B 054EFA7C
F400D27A 13B8794E 00C04118 66D0E5EB 1F94E3F9 5D11EE51 FCA02E1A 1E2F7F66
```

Group membership issuing process, by User (steps a to g)

s_i =

```
0008C392 9FB3FCE2 5F32AC12 C5453738 EAAB9A43 673FB71C C427CDE2 FEFD754B E342A391 F74C2FFF
```

u =

```
000D1D99 4C806081 1B393F57 CCC1487D 52C595DB 9C1E5944 5B60E119 357A3020 FEE55A5A 1433F1F0
```

v =

```
0005032F 9006BBE2 01404A42 9EB25D02 7BE56406 711EF075 C8A037E5 0915B769 C6444E3E 5B119DBB
```

k_s =

```
0000F98C 5BA21D6B E7044477 88683F64 73D42A00 8F4207D0 09FB4232 23E12FE4 86699DFB 9F195910
```

k_u =

```
0006285D 23FE1144 E5814EB1 ACC57970 0BA562EE 8793117A 4521502B 9C642A81 F5B6F960 16D83DB3
```

k_v =

```
000ADD31 D725F262 9F8D66CE DBF67DCA 07631DCE 6419CD3C 8E72AE66 FABA9B4A C4DA84A9 66AB9880
```

S_i =

```
09E197C1 A1A443F8 CF5D096D 068825C0 FAB8CF6F 0711774D AE144464 7B01DF1D E9A0C887 324846FD
506A2196 7F50905E 5356F705 A82B7863 55E503ED B3322A99 6AA5B54F 855C9E15 7938D924 7A337DE8
C00B3E18 2EB5BF8E B93D31E7 72A6D658 FD39998A 0C66AC02 7C23CD57 5214DDE6 C5390DA4
```

Y_i =

```
03D49577 C6AF338B 48426FBD 7E6CE9FD 5A4182EF 9ED7E918 480F2654 AC371009 8EB6B80C 2324D20C
199AD9BC 6E64AABE 66CD3110 7209481E D3310EB3 08BF825B EE04DDFD A4421C0B 6BD75F99 DBFA3249
36A1C472 10331473 7917A1C2 A3447E54 0F6ECE77 B539792D 75D634C3 270A1F81 365C7399 1491DC08
9D1FD04F F00BCD78 BE38D5F4 B622ABD2 2AFAE9E3 7D13460B 6EC83996 9ED25B4F 4489FDDC 842DB5AB
3DA80C29 58382773 03CD330A 04A90F95 6F75A534 FD770F9C C3E22255 1E48CECA A831C1AE 8401F695
791DCB4D ABAC7C45 BC358D70 5FF1EB56 30B1E8D1 FF553090 8F7A205A 8C4A551D
```

C_1 =

```
0175B2E3 B1EB724D AC333A2C 1489C823 180CE55C 456C0BEC 5F37CFC8 B2ED733F 9868CEA1 FDCF1456
BAFE2844 348BA116 1EF1BB01 229DD0E0 E6500498 2B0A0575 750035B8 0DE962B7 5CC504E9 CACCB39C
3ED11071 043C304D A109BF56 AC52FBD2 1D3831F5 E2E8A52C 22CDF9BE 9F2FFE08 E0553124 0FAE48A9
0400D301 3BBDB934 D5DDB75B 0916CA45 BBFB0023 8C1CFB0E 7E7E43B4 8BFA0557 8B7B0E29 435DDF32
80DC6BA0 95F2A821 2D1B84BA 0B44032F 49D8A37B 75D3FF31 FAA2615B D1EE1D63 B95EFCEE 74CC5E3B
9518DAC0 B837D198 D8EEC17F B8F833BA CC20CF26 FC115792 B32AC29C 7C2FC9B1
```

C_2 =

```
005339C6 B77BD37A E2DF9815 207556A0 148345D9 A92E2414 CE59322C 61CCBA78 2C6BE169 A564D146
E5F25152 F05D4B50 F6A3BF92 6277799D EA380425 0D677BFA 1EF76F9B D2BE6396 CEAFA24D 7DC5B4DD
B9A6B138 2F82B698 E6CD2F6C C2FA62F5 64D85E9E 0F4E4AC3 6E96A5BC 87F9CFB9 F66D8385 0602FAEC
8B1B5847 DA292C54 D2AB87D8 252B20FA 4115DF59 F7C44FB9 653C413E C89250B3 C2D1205A E3E07CC3
4F6EB207 8509EE9B 091AFF92 CF330258 F89DE671 956452B4 9D6EB824 8EF3E5B1 BC712B81 A4FCEAB5
A60A5AF7 E5D3648F 814EF6B3 D1AA23A4 360011EA F2A547C5 ED39C887 13CD87B8
```

C_3 =

```
0A5E69BC AEC67CE6 71F7CD7C 805304F0 EDB58BD4 B7056C2D 6E543456 124E9C8D EBD8781A 81C36781
2A49B85C FCE992FF 04666CF2 4445010A 4F8706AC 6E1B4FFD 991DCA6A 44ACB8CE ADD91ABF CE9E0CEF
32A2D0D5 BDA1AC6E B1FACC82 9AF77262 D4C2FE6D 9DBAB456 BDD66F8E 2DDEF704 9111ECB9 125D7AB2
FCD09135 CC19DE72 75A19CFE BAE23FA2 48257594 A20AC389 8CBFEAAA E063AE80 3ADD7E47 0FB6C630
41F8A599 C5CBBD82 DF62FB11 0DC8120C 05A05B2E 3E10A298 70EE2209 C2D231CE 0D45B547 DD0E023F
E88AA5B4 7258F1A3 150B1CC0 A0250181 D517001C 91FA87DF 1D0DB67D 1D2A777A
```

C_4 =

```
11AE9ABC 48ED1B64 60920F55 6F84D71D 3C269BE9 0C5B6D02 6179D68E EB3FE116 7935EF1B E7251890
971E8A65 FAA59723 DC8EB4B4 A4319FCC 42C406B8 05B309A4 B4818C7E CF56E26C 9B44A562 DDEF16C3
9DCDA1FD 1497B3DD E6D07284 402B92D4 EAF7131A A01811DF 65322530 23E29528 1F08DC0C 10B1D24B
3C637D74 C06F9CD8 81087E86 41116907 9191C8CC EC96BEB4 958F5B41 B65EE0B3 63B0AFDF C7B993A8
C0DF3C82 F43AA50E EAFD0E0E F19A11C7 14CE09DF 2E001328 EE43D423 00BAB8ED 09EA0031 4A798004
B379CA09 52E37AB7 DE7EA61F 3B5A99C9 1065166B DBE3B8D7 4CF2C563 0CDDB3A9
```

K =

```
154ABC6A 027C2127 9294EE43 808865FE 0E70EFA3 00851AA5 9CEE1412 4F4B03A4 9C911827 E286E624
917C23CC 9F654C4C 7754ED33 4188073F C537014C 72115AD4 6DF73552 55568884 4CE53229 91F77EAF
9BC60EEA 29BAEBE6 55AF5367 5B50CE1A FDAD557A D588D7DE 468E0CAD 665D6D8B 876A0F20
```

K_1 =

```
0070C20D 19040ACD FB18F89E 2E46E680 BA888841 1DE2D5F8 05D36FA0 351B54A0 86B7281A 31D3087B
B0680E5B 34629EA2 4A91BB6A 38E78913 3E1702D0 50FB66D9 304E2B45 E5BC69ED 6A17BF67 D6D28CD9
```

```
DDF18E24 5CD47165 28673F7E FF62BDA6 3E8C5440 3FCF3419 EF34EEF9 9784782B 648D1DBE 0ECF1D5A
87867CB1 8EA60C89 87F606C8 FFEB73CA 487F0869 40D69D81 A128A0AD 7DE6F9BF 84E1874B F9D9DFD5
5EDFA6E9 5CA4D884 573B4097 00980D45 B240DB7F F2659CEA 44872D91 77A3A034 AF996742 D8AEB0C7
3E2D9C95 894FFA81 23CEAB2D BEE9562B 20D034DC 8CAA6290 9CF26051 2587023E
```

K_2 =

```
112C496E E5A510B9 460C804B F3B4A3EC 2E2554D0 3C2C32A1 6D76224D DE8B51B8 B0AF1DCA 1C949114
30CAA5F0 D6D2FDA8 D7FC7558 017F665E 523A05DE A85163B2 2DF0DCD4 847291D6 FADA08F4 996F1DA5
3DC920A7 DAC5AA41 71273360 D88796C2 D759372D A39AD3D2 DD02A3DF 330CF7CD FB2C58F4 01BFC29E
72560D30 399D1940 CDB7CB02 291C2C0D FA3C26B9 D39BC1B6 1720B12A DD1D9448 87CE43E2 52422AE9
6AEC52AC F6365707 CD89F88D D5F30AEB F59EFB10 2403403E F7DD9C8A 71091523 04169289 23EFDDBB
C3029064 FBD3D850 E07A8C58 B94C2C47 F6A0CA5C 3D6C8D50 9D137D0C 8CB09A96
```

K_3 =

```
03923388 FB35977C 9FE41ED8 F2391065 D6316795 AFA4729A FC312E77 26C6B78E 6EE33979 BB579497
4281F9D8 B48B81B9 788AC1EE 73DE9308 8CC80062 05BBF7EB BD70212E C83DC3E2 1ED6C78C 6605DE39
6E702E99 71C80C4A AB15C2DD 267B51D4 E6532E03 F79280E6 0B5B34BD 07876662 AB1724CD 0372DAE7
D9EDEFC1 4E37BFF6 13024E06 19FC9519 AD1B9353 3ED696B8 AA80CDEC 47729A01 3085573B ACC0F993
DD4EA014 6E2F3BBD F620CB0E DB5F0EF9 43202701 05D0C863 BBA7A148 2B248266 69A4C326 3C494294
8A409AB5 6F0A29F7 41F5C9BF 1E691DC5 F3ECC43D A2495C8A C3C17A1C D0A1D7FF
```

K_4 =

```
14CA8575 7ECFC836 A11A41A7 441F391D CE38902E 1D24C3AF 6BDD93C8 C8EF38D9 821F3101 B16ECD77
10E350CF D03050F9 EE84E35A 37DAFA81 C9A81535 7610CC98 7AF5535D 9D0C2082 03C936F9 E463381A
430C9738 A85CFB3F 80C4BD64 621E48A7 011C2240 6151C7F9 551579CC 74934491 239AECE2 0FAB80CE
54D55DB1 EDDA96BC 73A67680 71329A60 037F414C AA682C30 08068B76 70CA532D 6DFAFE4C 82D48289
A5227DCF 3EBB999E 0BE310D8 B8961242 7C156295 D4BADCFE 1F340D24 7E057DBA A733E3BD E15B465F
A04264BA 48E89FE9 A14E7103 A70A55AE EB5F97AC D8B82CD5 85C15648 A36CD537
```

c =

```
C21CC732 71BC695B C9D6B2A5 DB58BC2B E18D3EA2 D5880474 15B12880 FA7D3710
```

z_s =

```
0006A531 52D0EEE2 4329787C FACA9404 18ECE765 DBD3E507 D7270767 2D018D44 339809CB 12101E1C
63A6C34C 9A1043E1 62EA7D1D E56FAE21 58105132 EFBC6C85 1D6531CE E7AF2200
```

z_u =

```
0009F1E7 9D556D4C EE0827A0 3F86A966 22FD79B9 6ACE239D 96496E3B 11B7CE15 BED7A36C C581BFE4
52322ABD 338728B5 09B2A623 A4B5464F BA12081D 40F38F43 6EE308FA 0541ECB3
```

z_v =

```
0003CCFA 4ACFAD75 94A391FC 071DB29D F13A6473 0B057E29 51DC3F79 96B881CF 33321F95 A922FCE8
93A44DC3 61585307 7DC56852 8F925C30 62818382 0282FA85 9E0F8F42 82F7A130
```

Group membership issuing process, by Issuer (steps h to o)

K' =

```
154ABC6A 027C2127 9294EE43 808865FE 0E70EFA3 00851AA5 9CEE1412 4F4B03A4 9C911827 E286E624
917C23CC 9F654C4C 7754ED33 4188073F C537014C 72115AD4 6DF73552 55568884 4CE53229 91F77EAF
9BC60EEA 29BAEBE6 55AF5367 5B50CE1A FDAD557A D588D7DE 468E0CAD 665D6D8B 876A0F20
```

$K_1^{'}$ =

```
0070C20D 19040ACD FB18F89E 2E46E680 BA888841 1DE2D5F8 05D36FA0 351B54A0 86B7281A 31D3087B
B0680E5B 34629EA2 4A91BB6A 38E78913 3E1702D0 50FB66D9 304E2B45 E5BC69ED 6A17BF67 D6D28CD9
DDF18E24 5CD47165 28673F7E FF62BDA6 3E8C5440 3FCF3419 EF34EEF9 9784782B 648D1DBE 0ECF1D5A
87867CB1 8EA60C89 87F606C8 FFEB73CA 487F0869 40D69D81 A128A0AD 7DE6F9BF 84E1874B F9D9DFD5
5EDFA6E9 5CA4D884 573B4097 00980D45 B240DB7F F2659CEA 44872D91 77A3A034 AF996742 D8AEB0C7
3E2D9C95 894FFA81 23CEAB2D BEE9562B 20D034DC 8CAA6290 9CF26051 2587023E
```