INTERNATIONAL STANDARD

ISO/IEC 11770-4

Second edition
2017-11-15

AMENDMENT 1
2019-09

Information technology — Security techniques — Key management —

Part 4: Mechanisms based on weak secrets

AMENDMENT 1: Unbalanced Password-Authenticated Key Agreement with Identity-Based Cryptosystems (UPAKA-IBC)

Technologies de l'information — Techniques de sécurité — Gestion de clés —

Partie 4: Mécanismes basés sur des secrets faibles

AMENDEMENT 1: Accord dissymétrique de clé authentifié par mot de passe utilisant un mécanisme de chiffrement basé sur l'identité

Reference number
ISO/IEC 11770-4:2017/Amd.1:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
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).

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

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.

This document was prepared by Joint 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.

Information technology — Security techniques — Key management —

Part 4: Mechanisms based on weak secrets

AMENDMENT 1: Unbalanced Password-Authenticated Key Agreement with Identity-Based Cryptosystems (UPAKA-IBC)

Introduction

Insert the following paragraph after Note 2:

d) **Unbalanced password-authenticated key agreement with identity-based cryptosystems:** Establish one or more secret keys for an entity, *A*, associated with another entity, *B*, where:

1) *A* and *B* share a weak secret and *B* has a strong secret; or

2) *A* has a weak secret and *B* has verification data derived from *A*'s weak secret with a one-way function along with a strong secret.

An example of a strong secret could be a long random key. This strong secret is independent of the weak secret and has been generated for an identity-based cryptosystem.

In an unbalanced password-authenticated key agreement mechanism with an identity-based cryptosystem, the shared secret keys are the result of a data exchange between the two entities. The shared secret keys are established if, and only if, *A* has used the weak secret and *B* has a strong secret corresponding to its identity and neither of the two entities can predetermine the values of the shared secret keys.

NOTE 3 This type of key agreement mechanism runs between entities with unbalanced security requirements such as in the client-server model. It is suitable for the case where a client (*A*) conducts authentication based on both a human-memorable password and a server (*B*)'s identity while enhancing server authentication (thus avoiding an attack on a cryptographic protocol in which an unauthorized party masquerades as a legitimate server, which is often called "a server impersonation attack").

Clause 3

Insert the following terminological entry at the end of the clause:

3.39
master-secret key
secret value used by the private key generator to compute private keys for an identity-based encryption (IBE) or identity-based signature (IBS) scheme

Clause 4

Insert the following abbreviated terms:

CT	ciphertext, an octet string
G_1	point of order r on E over $F(q)$, in the same subgroup as G
IBE	identity-based encryption
IBS	identity-based signature
ID	distinguished identifier that is uniquely assigned to an identity, which is an octet string
κ	security parameter
Msg	plaintext, an octet string
msk	master-secret key of IBE or IBS
parms	domain parameters of IBE or IBS
σ	signature, an octet string
sk_{ID}	private key corresponding to an entity with distinguished (octet) identifier *ID* for an IBE or IBS scheme

Clause 5

Replace the definition of G by the definition of G and G_1 as follows:

— G, G_1 curve points of order r (G and G_1 are called the generators of a subgroup of r points on E). It is assumed that the logarithm of G_1 to the base G is unknown;

Clause 5

Add the following paragraph at the end:

Annex B defines the object identifiers which shall be used to identify the mechanisms specified in this document.

Clause 8

Add new Clause 8 as follows:

8 Unbalanced password-authenticated key agreement with identity-based cryptosystems

8.1 General

This clause specifies two unbalanced password-authenticated key agreement mechanisms using identity-based cryptosystems. In these mechanisms, one entity has a weak secret derived from a password, and the other entity has the password verification data and a strong secret, i.e., a long random key which is used with an identity-based cryptosystem. An identity-based cryptosystem is an

asymmetric cryptographic technique that allows a public key to be defined as a string, such as an IP address or e-mail address, that represents an entity. The corresponding private key is calculated from an identity, a set of domain parameters, and a domain-wide secret value called a master-secret key. A user public key is calculated by anyone who has the necessary domain parameters, while the master secret key is needed to calculate a user private key, and the calculation can only be performed by a (trusted) private key generator that has this secret.

The two unbalanced password-authenticated key agreement mechanisms with identity-based cryptosystems have the following initialization process and key establishment process.

Initialization process

The two entities involved agree to use a set of valid domain parameters, a set of key derivation parameters and a set of functions, all of which can be publicly known.

The two entities also agree to use either a shared weak secret derived from a password or shared password-based information, i.e. one entity has a password-based weak secret and the other entity has the corresponding password verification element.

Key establishment process

a) *Generate and exchange key tokens.* Each of the two entities involved randomly chooses one or more key token factors associated with the domain parameters, creates the corresponding key tokens, which may be associated with the password or password verification element (a key token associated with the password or password verification element is called a "password-entangled key token"), and then makes the key tokens available to the other entity.

b) *Check validity of key tokens.* Depending on the operations for producing key tokens in step a), each of the two entities involved chooses an appropriate method to validate the received key tokens based on the domain parameters. If any validation fails, output "invalid" and stop.

c) *Derive shared secret keys.* Each of the two entities involved applies certain secret value derivation functions to their own key token factor, the other entity's key tokens and/or shared password or password verification element to produce a shared secret value. Each entity further applies a key derivation function to the shared secret value and the key derivation parameters, to derive one or more shared secret keys.

d) *Check key confirmation.* The two entities involved use the shared secret keys established using the above steps to confirm their awareness of the keys to each other. This step is optional in Unbalanced Key Agreement Mechanisms with Password and Identity-based Encryption and in Unbalanced Key Agreement Mechanisms with Password and Identity-based Signature.

For convenience, an unbalanced key agreement mechanism with password and identity-based encryption and an unbalanced key agreement mechanism with password and identity-based signature are denoted by UKAM-PiE and UKAM-PiS, respectively.

Because of the use of weak secrets, countermeasures against online guessing attacks shall be taken into account. For example, counters that indicate the number of failed connections can be introduced:

— a counter that tracks the total number of unsuccessful authentication trials in a row;

— a counter that tracks the total number of unsuccessful authentication events during the period of use of a specific password;

— a counter that tracks the total number of authentication events (successful and unsuccessful) during the period of use of the specific password.

NOTE In applications using a UKAM-PiE or UKAM-PiS, *A* can play the role of a client and *B* can play the role of a server.

8.2 Unbalanced Key Agreement Mechanism with Password and Identity-based Encryption (UKAM-PiE)

8.2.1 General

This mechanism is designed to achieve authenticated key agreement using a password and an identity-based encryption scheme, and it establishes one or more shared secret keys between entities *A* and *B*. In the mechanism, *A* and *B* share a password-based octet string π. *B* additionally has a strong secret, i.e., a decryption key which is used for an identity-based encryption scheme. This mechanism provides unilateral explicit key authentication and optionally mutual key authentication.

An identity-based encryption scheme consists of the following four algorithms.

— *IBE.Setup*(κ). Given a security parameter κ, generate a tuple <*parms*, *msk*>, where *parms* denotes IBE domain parameters, *msk* denotes a master-secret key.

— *IBE.Extract*(*parms*, *msk*, *ID*). Given *parms, msk,* and *ID*, generate a private key sk_{ID} for *ID*.

— *IBE.Enc*(*parms*, *ID*, *Msg*). Given *parms*, *ID*, and a plaintext *Msg*, perform the encryption and output the ciphertext of *Msg*, *CT,* for *ID*. Note that *Msg* and *CT* are octet strings.

— *IBE.Dec*(*parms*, *ID*, sk_{ID}, *CT*). Given *parms*, *ID*, and sk_{ID}, decrypt a ciphertext *CT* and output a plaintext or "error".

In general, the setup, key extraction and encryption algorithms are probabilistic, while the decryption algorithm is deterministic. It is recommended that applications establish a methodology for authenticating access to private keys by using the string *ID* as an identity in a trusted authentication system (see ISO/IEC 18033-5[33] for further guidance on identity-based ciphers).

This mechanism works in both the DL and EC settings.

As noted in ISO/IEC 18033-5, a general purpose IBE scheme should satisfy the appropriate security level, namely semantic security against an adaptive chosen ciphertext attack.

NOTE UKAM-PiE is based on Bibliographic reference [29].

8.2.2 Prior shared parameters

Key agreement between two entities *A* and *B* takes place in a shared context consisting of the following parameters:

— a set of valid domain parameters (either DL domain parameters or EC domain parameters) specified in Clause 5;

— a password-based octet string, π, used by *A* and *B;*

— a key token generation function, *D*;

— a private key, sk_B, used by *B*;

— an encrypted key token generation function, *EK*, used by *A*;

— a key token decryption function, *KD*, used by *B*;

— a password verification function, *S*, used by *B*;

— a key token check function, *T*;

— a secret value derivation function, V_S;

— a key derivation function, *K*;

— one or more key derivation parameter octet strings $\{P_1, P_2, ...\}$, where *A* and *B* shall agree to use the same values;

— the length of a shared secret key, L_K.

8.2.3 Functions

8.2.3.1 Key token generation function, *D*

The key token generation function, *D*, is as defined in 6.5.3.2.

8.2.3.2 Encrypted key token generation function, *EK*

The encrypted key token generation function, *EK*, takes the IBE domain parameters *parms*, an *ID*, a password-based octet string π, and an output *w* (or *W*) of the function *D*, and produces an encrypted key token *CT* as output, i.e. *CT* = *EK*(*parms, ID*, π, *w* (or *W*)). UKAM-PiE is used with the following function *EK*:

— Given the IBE domain parameters *parms* and three inputs, an octet string *ID*, an octet string π, and the output *w* (or *W*) of the function *D*, *EK* is defined as follows:

  — compute *d* = I2OS(BS2I(*H*(π)));

  — compute *e* = $GE2OS_X$(*w* (or *W*));

  — compute *CT* = *IBE.Enc*(*parms, ID, d*||*e*);

  — output *CT*.

Functions BS2I (Bit String to Integer conversion), I2OS (Integer to Octet String conversion) and $GE2OS_X$ (Group Element to Octet String conversion) are described in Annex A.

8.2.3.3 Key token decryption function, *KD*

The key token decryption, *KD*, takes the IBE domain parameters *parms*, an *ID*, a private key sk_{ID} for *ID*, an output *CT* of the function *EK*, and produces an ordered pair made up of an octet string *d* and a group element *w* (or *W*) as output. UKAM-PiE is used with either KD_{DL} or KD_{EC} as the function *KD*:

— KD_{DL} is suitable for use when the mechanism is used with the DL domain parameters, i.e. it operates over the multiplicative group of elements defined over *F*(*q*). Given the DL domain parameters (including *g* and *q*), the IBE domain parameters *parms*, an octet string *ID*, a group element sk_{ID}, and the output *CT* of the function *EK*, KD_{DL} is defined as follows:

  — compute *IBE.Dec*(*parms, ID,* sk_{ID}, *CT*);

  — if the output of *IBE.Dec* is "error", then output "error";

  — else, the output of *IBE.Dec* is *d*||*e* and compute *w* = I2FE(OS2I(*e*));

  — output (*d*, *w*).

— KD_{EC} is suitable for use when the mechanism is used with the EC domain parameters, i.e. it operates over the additive group of elements in an elliptic curve defined over *F*(*q*). Given the EC domain parameters (including *G*), the IBE domain parameters *parms*, an octet string *ID*, a group element sk_{ID}, and the output *CT* of the function *EK*, KD_{EC} is defined as follows:

  — compute *IBE.Dec*(*parms, ID,* sk_{ID}, *CT*);

  — if the output of *IBE.Dec* is "error", then output "error";

  — else, the output of *IBE.Dec* is *d*||*e* and compute *W* = I2FE(OS2I(*e*));

— output (d, W).

Functions OS2I (Octet String to Integer conversion) and I2FE (Integer to Field Element conversion) are described in Annex A.

8.2.3.4 Password verification function, S

The password verification function, S, takes an octet string d output by the function KD, and a password-based octet string π, and produces a Boolean value written $S(d, \pi)$ as output. UKAM-PiE is used with the following function S:

Given an octet string d output by the function KD and an octet string π, S is defined as follows:

— if d = I2OS(BS2I($H(\pi)$)), $S(d, \pi) = 1$;

— else, $S(d, \pi) = 0$.

Functions BS2I (Bit String to Integer conversion) and I2OS (Integer to Octet String conversion) are described in Annex A.

8.2.3.5 Key token check function, T

The key token check function, T, is as defined in 6.2.3.3.

8.2.3.6 Secret value derivation function, V_S

The secret value derivation function, V_S, takes an integer x and a selected group element y (or Y) as input and produces another group element written $V_S(x, y$ (or $Y))$ as output. UKAM-PiE is used with either V_{SDL} or V_{SEC} as the function V_S:

— V_{SDL} is suitable for use when the mechanism is used with the DL domain parameters, i.e. it operates over the multiplicative group of elements defined over $F(q)$. Given the DL domain parameters (including r and q), and two inputs, x from {1, ..., r − 1} and y from {2, ..., q − 2}, V_{SDL} is defined as in Formula (35):

$$V_{SDL}(x, y) = y^x \bmod q \tag{35}$$

— V_{SEC} is suitable for use when the mechanism is used with the EC domain parameters, i.e. it operates over the additive group of elements in an elliptic curve defined over $F(q)$. Given the EC domain parameters (including r), and two inputs, x from {1, ..., r − 1} and a point Y ($\neq 0_E$) on E, V_{SEC} is defined as in Formula (36):

$$V_{SEC}(x, Y) = \lfloor x \rfloor \times Y \tag{36}$$

8.2.3.7 Key derivation function, K

The key derivation function, K, is as defined in 6.2.3.6.

8.2.4 Key agreement operation

This mechanism involves both A and B performing a sequence of up to four steps, numbered A1-A4 and B1-B4 (for the steps to be followed by A and B, respectively). Steps A3, A4, B3, and B4 are optional.

Encrypted key token construction (A1)

A performs the following steps:

— choose an integer s_A randomly from {1, ..., r − 1} as its key token factor;

— compute $w_A = D(s_A)$ as its key token;

— compute $CT = EK(parms, ID_B, \pi, w_A)$ as its encrypted key token;

— make CT available to B (by sending (ID_A, CT) from A to B).

Key token construction (B1)

B performs the following steps:

— receive CT from A;

— compute $(d, w_A) = KD(parms, ID_B, sk_B, CT)$;

— if the output of KD is "error", then output "invalid" and stop; otherwise, continue;

— check validity of w_A using $T(w_A)$: if $T(w_A) = 0$, output "invalid" and stop; otherwise, continue;

— check validity of d using $S(d, \pi)$: if $S(d, \pi) = 0$, output "invalid" and stop; otherwise, continue;

— choose an integer s_B randomly from $\{1, \ldots, r - 1\}$ as its key token factor;

— compute $w_B = D(s_B)$ as its key token;

— make w_B available to A (by sending (ID_B, w_B) from B to A).

Shared secret key derivation (A2)

A performs the following steps:

— receive w_B from B;

— check validity of w_B using $T(w_B)$: if $T(w_B) = 0$, output "invalid" and stop; otherwise, continue;

— compute $z = V_S(s_A, w_B)$ as a shared secret value;

— compute $K_i = K(ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z), P_i, L_K)$ as a shared secret key for each key derivation parameter, P_i.

Shared secret key derivation (B2)

B performs the following steps:

— compute $z = V_S(s_B, w_A)$ as a shared secret value;

— compute $K_i = K(ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z), P_i, L_K)$ as a shared secret key for each key derivation parameter, P_i.

Key confirmation (A3 and B3) (optional)

A performs the following steps (A3):

— compute $o_A = H(\text{I2OS}(4)||ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z))$;

— make o_A available to B.

B performs the following steps (B3):

— receive o_A from A;

— compute $o_A' = H(\text{I2OS}(4)||ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z))$;

— check if $o_A \neq o_A'$, output "invalid" and stop.

Key confirmation (B4 and A4) (optional)

B performs the following steps (B4):

— compute $o_B = H(\text{I2OS}(3)||ID_B||ID_A||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z))$;

— make o_B available to *A*.

A performs the following steps (A4):

— receive o_B from *B*;

— compute $o_B' = H(\text{I2OS}(3)||ID_B||ID_A||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z))$;

— check if $o_B \neq o_B'$, output "invalid" and stop.

Functions I2OS (Integer to Octet String conversion) and GE2OS_X (Group Element to Octet String conversion) are described in Annex A.

NOTE 1 A group element in this mechanism is a point on the curve *E* in the EC setting, or an integer in the range {1, …, $q - 1$} in the DL setting.

NOTE 2 This mechanism can provide privacy protection for the client if the identifier of client is encrypted along with the password and ephemeral key using identity-based encryption.

NOTE 3 In the shared secret key derivations (A2 and B2), a shared key can alternatively be generated using π as an additional input to function *K*. It prevents a client impersonation attack when an ephemeral private exponent s_A is compromised but π is not revealed, that is, without doing offline brute force password attacks.

8.3 Unbalanced Key Agreement Mechanism with Password and Identity-based Signature (UKAM-PiS)

8.3.1 General

This mechanism is designed to achieve authenticated key agreement using a password and an identity-based signature scheme, and it establishes one or more shared secret keys between entities *A* and *B*. In the mechanism, *A* has a password-based octet string π, and *B* has password verification element *v* corresponding to π. *B* additionally has a strong secret, i.e., a signing key for an identity-based signature scheme. This mechanism provides mutual implicit key authentication and, optionally, mutual explicit key authentication.

An identity-based signature scheme (see Bibliographic references [28], [31], [32] and [34]) consists of the following four algorithms.

— *IBS.Setup*(κ). Given a security parameter κ, generate a tuple <*parms*, *msk*>, where *parms* denotes IBS domain parameters and *msk* denotes a master-secret key.

— *IBS.Extract*(*parms*, *msk*, *ID*). Given *parms*, *msk*, and *ID*, generate a private key sk_{ID} for *ID*.

— *IBS.Sign*(*parms*, *ID*, sk_{ID}, *Msg*). Given *parms*, *ID*, sk_{ID}, and a message *Msg*, perform the signing process and output a signature, σ, for *ID*. Note that *Msg* and σ are octet strings.

— *IBS.Verify*(*parms*, *ID*, σ, *Msg*). Given *parms*, *ID*, a signature σ, and *Msg*, perform the verifying process and output 1 if it holds, or 0 otherwise.

In general, the setup, key extraction and signing algorithms are probabilistic, while the verification algorithm is deterministic. In an IBS setup, a trusted third party generates a master-secret key and the corresponding IBS domain parameters by invoking the *IBS.Setup* algorithm. The trusted third party makes the IBS domain parameters public, keeping the master-secret key private. When a signer requests a private key, a private key generator invokes the *IBS.Extract* algorithm and issues its output

to the signer as the private key via a secure channel. A signer generates a signature by running the *IBS.Sign* algorithm with the signer's private key. A verifier verifies the signature by running *IBS.Verify* with the signer's identity.

The signature verification is done without requiring interactions between a verifier and a signer, either directly or through a proxy such as a directory or certificate server, before verifying the signatures.

UKAM-PiS works in both the DL and EC settings.

NOTE This mechanism is based on Bibliographic reference [30] where the mechanism is called the simplified PWIBS-AKE. The notation here has been changed to conform with the notation used in this document.

8.3.2 Prior shared parameters

Key agreement between two entities A and B takes place in a shared context consisting of the following parameters:

— a set of valid domain parameters (either DL domain parameters or EC domain parameters) specified in Clause 5;

— a password-based octet string, π, used by A;

— a password verification element, $v = J(\pi)$ used by B, where J is a password verification element derivation function;

— a password-entangled key token generation function, C, used by A;

— a key token generation function, D, used by A;

— a signed key token generation function, SD, used by B;

— a key token check function, T;

— two secret value derivation functions, V_A and V_B, one for each entity;

— a key derivation function, K;

— one or more key derivation parameter octet strings $\{P_1, P_2, ...\}$, where A and B shall agree to use the same P_i value;

— the length of a shared secret key, L_K.

8.3.3 Functions

8.3.3.1 Password verification element derivation function, J

The password verification element derivation function, J, takes a password-based octet string π, as input and produces a selected group element defined over $F(q)$ written $J(\pi)$ as output. UKAM-PiS is used with either J_{DL} or J_{EC} as the function J:

— J_{DL} is suitable for use when the mechanism is used with the DL domain parameters, i.e. it operates over the multiplicative group of elements defined over $F(q)$. Given the DL domain parameters (including g_1 and q), and a password-based octet string π, J_{DL} is defined as in Formula (37):

$$J_{DL}(\pi) = g_1^{-\mathrm{BS2I}(H(\pi))} \bmod q \tag{37}$$

— J_{EC} is suitable for use when the mechanism is used with the EC domain parameters, i.e. it operates over the additive group of elements in an elliptic curve defined over $F(q)$. Given the EC domain parameters (including G_1), and a password-based octet string π, J_{EC} is defined as in Formula (38):

$$J_{EC}(\pi) = \left[-\text{BS2I}\left(H(\pi)\right)\right] \times G_1 \tag{38}$$

Function BS2I (Bit String to Integer conversion) is described in Annex A.

8.3.3.2 Key token generation function, *D*

The key token generation function, *D*, is as defined in 6.5.3.2.

8.3.3.3 Signed key token generation function, *SD*

The signed key token generation function, *SD*, takes the domain parameters of an identity-based signature scheme *parms*, an identity *ID* of a signer, a signing key *sk*, and an output *y* of the function *D* as input, and produces a signed key token written *SD*(*parms*, *ID*, *sk*, *y*) as output. UKAM-PiS is used with the following function *SD*:

Given the set of domain parameters of an identity-based signature scheme *parms*, an identity *ID* of a signer, a signing key *sk*, and an output *y* of the function *D*, *SD* is defined as in Formula (39):

$\sigma = IBS.Sign(parms, ID, sk, \text{GE2OS}_X(y))$

$$SD(parms, ID, sk, y) = (ID, y, \sigma) \tag{39}$$

8.3.3.4 Password-entangled key token generation function, *C*

The password-entangled key token generation function, *C*, takes two inputs, an integer *x* from {1, …, *r* − 1}, a password-based octet string π, and produces a selected group element written $C(x, \pi)$ as output. UKAM-PiS is used with either C_{DL} or C_{EC} as the function *C*:

— C_{DL} is suitable for use when the mechanism is used with the DL domain parameters, i.e. it operates over the multiplicative group of elements defined over $F(q)$. Let "*" denote the multiplicative operation of the group. Given the DL domain parameters (including g, $g_1$and q), and two inputs, x from {1, …, r − 1}, and a password-based octet string π, C_{DL} is defined as follows:

  — compute $e = \text{BS2I}(H(\pi))$;

  — compute $C_{DL}(x, \pi) = g^x * g_1^e \bmod q$;

  — check if $C_{DL}(x, \pi)$ is 1 or q − 1, output "invalid" and stop; otherwise, output $C_{DL}(x, \pi)$.

— C_{EC} is suitable for use when the mechanism is used with the EC domain parameters, i.e. it operates over the additive group of elements in an elliptic curve defined over $F(q)$. Given the EC domain parameters (including G, G_1, and r), and two inputs, x from {1, …, r − 1} and a password-based octet string π, C_{EC} is defined as follows:

  — compute $e = \text{BS2I}(H(\pi))$;

  — compute $C_{EC}(x, \pi) = [x] \times G + [e] \times G_1$;

  — check if $[4] \times C_{EC}(x, \pi) = 0_E$, output "invalid" and stop; otherwise output $C_{EC}(x, \pi)$.

Function BS2I (Bit String to Integer conversion) is described in Annex A.

8.3.3.5 Key token check function, T

The key token check function, T, as defined in 6.2.3.3.

8.3.3.6 Secret value derivation functions, V_A and V_B

a) The secret value derivation function, V_A, takes two inputs, an integer x_A from {1, ..., r – 1} and an output y_B (or Y_B) of function D, and produces a selected group element written $V_A(x_A, y_B$ (or Y_B)) as output.

b) The secret value derivation function, V_B, takes three inputs, an output v of function J, an integer x_B from {1, ..., r – 1}, an output w_A (or W_A) of function C, and produces a selected group element written $V_B(v, x_B, w_A$ (or W_A)) as output.

c) V_A and V_B satisfy the condition $V_A(x_A, y_B$ (or Y_B)) = $V_B(v, x_B, w_A$ (or W_A)).

UKAM-PiS is used with either V_{ADL} or V_{AEC} as the function V_A, and either V_{BDL} or V_{BEC} as the function V_B:

a) V_{ADL} is suitable for use when the mechanism is used with the DL domain parameters, i.e. it operates over the multiplicative group of $F(q)$. Given the DL domain parameters (including r and q), an integer x_A from {1, ..., r – 1} and an integer y_B from {2, ..., q – 2}, V_{ADL} is defined as follows:

   — compute $V_{ADL}(x_A, y_B) = y_B^{x_A} \bmod q$;

   — output $V_{ADL}(x_A, y_B)$.

b) V_{BDL} is suitable for use when the mechanism is used with the DL domain parameters, i.e. it operates over the multiplicative group of $F(q)$. Let "*" denote the multiplicative operation of the group. Given the DL domain parameters (including r and q), an output v_{DL} of function J_{DL}, an integer x_B from {1, ..., r – 1}, and an integer w_A from {2, ..., q – 2}, V_{BDL} is defined as follows:

   — compute $V_{BDL}(v_{DL}, x_B, w_A) = (w_A * v_{DL})^{x_B} \bmod q$;

   — output $V_{BDL}(v_{DL}, x_B, w_A)$.

c) V_{AEC} is suitable for use when the mechanism is used with the EC domain parameters, i.e. it operates over the additive group of elements in an elliptic curve defined over $F(q)$. Given the EC domain parameters (including r), an integer x_A from {1, ..., r – 1}, and a point Y_B (≠ 0_E) on E, V_{AEC} is defined as follows:

   — compute $V_{AEC}(x_A, Y_B) = [x_A] \times Y_B$;

   — output $V_{AEC}(x_A, Y_B)$.

d) V_{BEC} is suitable for use when the mechanism is used with the EC domain parameters, i.e. it operates over the additive group of elements in an elliptic curve defined over $F(q)$. Given the EC domain parameters (including r), an output v_{EC} of function J_{EC}, an integer x_B from {1, ..., r – 1}, and a point W_A (≠ 0_E) on E, V_{BEC} is defined as follows:

   — compute $V_{BEC}(v_{EC}, x_B, W_A) = [x_B] \times (W_A + v_{EC})$;

   — output $V_{BEC}(v_{EC}, x_B, W_A)$.

8.3.3.7 Key derivation function, K

The key derivation function, K, is as defined in 6.2.3.6.

8.3.4 Key agreement operation

This mechanism involves both A and B performing a sequence of up to four steps, numbered A1 to A4 and B1 to B4 (for the steps to be followed by A and B, respectively). Steps A3, B3, A4 and B4 are optional.

Password-entangled key token construction (A1)

A performs the following steps:

— choose an integer x_A randomly from {1, …, r – 1} as its key token factor;

— compute $w_A = C(x_A, \pi)$ as its key token;

— make w_A available to *B* (by sending (ID_A, w_A) from *A* to *B*).

Signed key token construction (B1)

B performs the following steps:

— choose an integer x_B randomly from {1, …, r – 1} as its key token factor;

— compute $y_B = D(x_B)$;

— compute $w_B = SD(parms, ID_B, sk_B, y_B) = (ID_B, y_B, \sigma_B)$ as its signed key token;

— make w_B available to *A* (by sending (ID_B, y_B, σ_B) from *B* to *A*).

Shared secret key derivation (A2)

A performs the following steps:

— receive $w_B = (ID_B, y_B, \sigma_B)$ from *B*;

— check the validity of w_B using $IBS.Verify(parms, ID_B, \sigma_B, \text{GE2OS}_X(y_B))$: if $IBS.Verify(parms, ID_B, \sigma_B, \text{GE2OS}_X(y_B)) = 0$, output "invalid" and stop; otherwise, continue;

— compute $z = V_A(x_A, y_B)$ as an agreed secret value;

— compute $K_i = K(ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(y_B)||\text{GE2OS}_X(z), P_i, L_K)$ as a shared secret key for each key derivation parameter, P_i.

Shared secret key derivation (B2)

B performs the following steps:

— receive w_A from *A*;

— compute $z = V_B(v, x_B, w_A)$ as an agreed secret value;

— compute $K_i = K(ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(y_B)||\text{GE2OS}_X(z), P_i, L_K)$ as a shared secret key for each key derivation parameter, P_i.

Key confirmation (A3 and B3) (optional)

A performs the following steps (A3):

— compute $o_A = H(\text{I2OS}(4)||ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(y_B)||\text{GE2OS}_X(\sigma_B)||\text{GE2OS}_X(z))$;

— make o_A available to *B*.

B performs the following steps (B3):

— receive o_A from *A*;

— compute $o_A' = H(\text{I2OS}(4)||ID_A||ID_B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(y_B)||\text{GE2OS}_X(\sigma_B)||\text{GE2OS}_X(z))$;

— check if $o_A \neq o_A'$, output "invalid" and stop.