INTERNATIONAL STANDARD

ISO/IEC 11770-4

First edition
2006-05-01

Information technology — Security techniques — Key management —

Part 4: Mechanisms based on weak secrets

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

Partie 4: Mécanismes basés sur des secrets faibles

Reference number
ISO/IEC 11770-4:2006(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.

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.

ISO/IEC 11770-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 11770 consists of the following parts, under the general title *Information technology — Security techniques — Key management*:

— *Part 1: Framework*

— *Part 2: Mechanisms using symmetric techniques*

— *Part 3: Mechanisms using asymmetric techniques*

— *Part 4: Mechanisms based on weak secrets*

Further parts may follow.

Information technology — Security techniques — Key management —

Part 4: Mechanisms based on weak secrets

1 Scope

This part of ISO/IEC 11770 defines key establishment mechanisms based on weak secrets, i.e., secrets that can be readily memorized by a human, and hence secrets that will be chosen from a relatively small set of possibilities. It specifies cryptographic techniques specifically designed to establish one or more secret keys based on a weak secret derived from a memorized password, while preventing off-line brute-force attacks associated with the weak secret. More specifically, these mechanisms are designed to achieve one of the following three goals.

1) **Balanced password-authenticated key agreement:** Establish one or more shared secret keys between two entities that share a common weak secret. In a balanced password-authenticated key agreement mechanism, 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 the two entities have used the same weak secret, and neither of the two entities can predetermine the values of the shared secret keys.

2) **Augmented password-authenticated key agreement:** Establish one or more shared secret keys between two entities *A* and *B*, where *A* has a weak secret and *B* has verification data derived from a one-way function of *A*'s weak secret. In an augmented password-authenticated key agreement mechanism, 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 the two entities have used the weak secret and the corresponding verification data, and neither of the two entities can predetermine the values of the shared secret keys.

   NOTE – This type of key agreement mechanism is unable to protect *A*'s weak secret being discovered by *B*, but only increases the cost for an adversary to get *A*'s weak secret from *B*. Therefore it is normally used between a client (*A*) and a server (*B*).

3) **Password-authenticated key retrieval:** Establish one or more secret keys for an entity, *A*, associated with another entity, *B*, where *A* has a weak secret and *B* has a strong secret associated with *A*'s weak secret. In an authenticated key retrieval mechanism, the secret keys, retrievable by *A* (not necessarily derivable by *B*), are the result of a data exchange between the two entities, and the secret keys are established if and only if the two entities have used the weak secret and the associated strong secret. However, although *B*'s strong secret is associated with *A*'s weak secret, the strong secret does not (in itself) contain sufficient information to permit either the weak secret or the secret keys established in the mechanism to be determined.

   NOTE – This type of key retrieval mechanism is used in those applications where *A* does not have secure storage for a strong secret, and requires *B*'s assistance to retrieve the strong secret for her. It is normally used between a client (*A*) and a server (*B*).

This part of ISO/IEC 11770 does not cover aspects of key management such as

— lifecycle management of weak secrets, strong secrets and established secret keys;

— mechanisms to store, archive, delete, destroy, etc. weak secrets, strong secrets, and established secret keys.

NOTE – The keys generated or retrieved through the use of weak secrets cannot be more secure against exhaustion than the sum of the weak secrets themselves. With this proviso, the mechanisms specified in this part of ISO/IEC 11770 are recommended for practical use in low-security environments.

2 Normative references

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

ISO/IEC 10118-3:2004, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*

ISO/IEC 11770-1:1996, *Information technology — Security techniques — Key management — Part 1: Framework*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
augmented password-authenticated key agreement
password-authenticated key agreement where entity *A* uses a password-based weak secret and entity *B* uses verification data derived from a one-way function of *A*'s weak secret to negotiate and authenticate one or more shared secret keys

3.2
balanced password-authenticated key agreement
password-authenticated key agreement where two entities *A* and *B* use a shared common password-based weak secret to negotiate and authenticate one or more shared secret keys

3.3
brute-force attack
attack on a cryptosystem that employs an exhaustive search of a set of keys, passwords or other data

3.4
collision-resistant hash-function
hash-function satisfying the following property: it is computationally infeasible to find any two distinct inputs which map to the same output

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

[ISO/IEC 10118-1:2000]

3.5
dictionary attack (on a password-based system)
attack on a cryptosystem that employs a search of a given list of passwords

NOTE – A dictionary attack on a password-based system can use a stored list of specific password values or a stored list of words from a natural language dictionary.

3.6
domain parameter
data item which is common to and known by or accessible to all entities within the domain

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

[ISO/IEC 9796-3:2000]

3.7
explicit key authentication from *A* to *B*
assurance for entity B that A is the only other entity that is in possession of the correct key

NOTE - Implicit key authentication from *A* to *B* and key confirmation from *A* to *B* together imply explicit key authentication from *A* to *B*.

[ISO/IEC 11770-3:1999]

3.8
hash-function
function which maps strings of bits to fixed-length strings of bits, satisfying the following two properties.

— It is computationally infeasible to find for a given output, an input which maps to this output.

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

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

[ISO/IEC 10118-1:2000]

3.9
hashed password
result of applying a hash-function to a password

3.10
implicit key authentication from *A* to *B*
assurance for entity *B* that *A* is the only other entity that can possibly be in possession of the correct key

[ISO/IEC 11770-3:1999]

3.11
key
sequence of symbols that controls the operation of a cryptographic transformation (e.g. encipherment, decipherment, cryptographic check function computation, signature calculation, or signature verification)

[ISO/IEC 11770-3:1999]

3.12
key agreement
process of establishing a shared secret key between entities in such a way that neither of them can predetermine the value of that key

[ISO/IEC 11770-1:1996]

3.13
key confirmation from *A* to *B*
assurance for entity *B* that entity *A* is in possession of the correct key

[ISO/IEC 11770-3:1999]

3.14
key control
ability to choose the key, or the parameters used in the key computation

[ISO/IEC 11770-1:1996]

3.15
key derivation function
function that utilizes shared secrets and other mutually known parameters as inputs, and outputs one or more shared secrets, which can be used as keys

3.16
key establishment
process of making available a shared secret key to one or more entities; key establishment includes key agreement, key transport and key retrieval

3.17
key management
administration and use of the generation, registration, certification, deregistration, distribution, installation, storage, archiving, revocation, derivation and destruction of keying material in accordance with a security policy

[ISO/IEC 11770-1:1996]

3.18
key retrieval
process of establishing a key for one or more entities known as the retrieving entities with the involvement of one or more other entities who are not necessarily able to access the key after the process, and which normally requires authentication of the retrieving entity/entities by the other entity/entities

3.19
key token
key establishment message sent from one entity to another entity during the execution of a key establishment mechanism

3.20
key token check function
function that utilizes a key token and other publicly known parameters as input, and outputs a Boolean value during the execution of a key establishment mechanism

3.21
key token factor
value that is kept secret and that is used, possibly in conjunction with a weak secret, to create a key token

3.22
key token generation function
function that utilizes a key token factor and other parameters as input, and outputs a key token during the execution of a key establishment mechanism

3.23
mutual key authentication
assurance for two entities that only the other entity can possibly be in possession of the correct key

3.24
one-way function
function with the property that it is easy to compute the output for a given input but it is computationally infeasible to find for a given output an input which maps to this output

[ISO/IEC 11770-3:1999]

3.25
password
secret word, phrase, number or character sequence used for entity authentication, which is a memorized weak secret

3.26
password-authenticated key agreement
process of establishing one or more shared secret keys between two entities using prior shared password-based information (which means that either both of them have the same shared password or one has the password and the other has password verification data) and neither of them can predetermine the values of the shared secret keys

3.27
password-authenticated key retrieval
key retrieval process where one entity *A* has a weak secret derived from a password, and the other entity *B* has a strong secret associated with *A*'s weak secret; these two entities, using their own secrets, negotiate a secret key which is retrievable by *A*, but not (necessarily) derivable by *B*

3.28
password-entangled key token
key token which is derived from both a weak secret and a key token factor

3.29
password verification data
data that is used to verify an entity's knowledge of a specific password

3.30
random element derivation function
function that utilizes a password and other parameters as input, and outputs a random element

3.31
salt
random variable incorporated as secondary input to a one-way or encryption function that is used to derive password verification data

3.32
secret
value known only to authorized entities

3.33
secret value derivation function
function that utilizes a key token factor, a key token and other parameters as input, and outputs a secret value, which is used to compute one or more secret keys

3.34
secret key
key used with symmetric cryptographic techniques by a specified set of entities

[ISO/IEC 18033-1:2005]

3.35
strong secret
secret with a sufficient degree of entropy that conducting an exhaustive search for the secret is infeasible, even given knowledge that would enable a correct guess for the secret to be distinguished from an incorrect guess

NOTE – This might, for example, be achieved by randomly choosing the secret from a sufficiently large set of possible values with an even probability distribution.

3.36
weak secret
secret that can be conveniently memorized by a human being; typically this means that the entropy of the secret is limited, so that an exhaustive search for the secret may be feasible, given knowledge that would enable a correct guess for the secret to be distinguished from an incorrect guess

4 Symbols and notation

For the purposes of this document, the following symbols and notation apply.

a_1, a_2	elliptic curve coefficients
A, B	distinguishing identifiers of entities
b, b_i	bits (i.e. either 0 or 1)
BS2I	a function that converts a bit string into an integer
c	an integer satisfying $1 \leq c \leq q - 1$
C,C_{DL},C_{EC}	functions for generating a key token based on a password and a key token factor
D,D_{DL},D_{EC}	functions for generating a key token based on only a key token factor
E	an elliptic curve defined by two elliptic curve coefficients, a_1 and a_2
$F(q)$	the finite field of cardinality q
FE2I	a function that converts a field element into an integer
FE2OS	a function that converts a field element into an octet string
g,g_1,g_a,g_b	elements of multiplicative order r in $F(q)$
G,G_a,G_b	points of order r on E over $F(q)$
g_{q-1}	an element of multiplicative order q -1 in $F(q)$
$GE2OS_X$	a function that converts a group element into an octet string; when the group element is a point on E, this function converts the x-coordinate of the point into an octet string and ignores the y-coordinate
H	a hash-function taking an octet string as input and giving a bit string as output, e.g. one of the dedicated hash-functions specified in ISO/IEC 10118-3
$h(x, L_K)$	a hash-function taking an octet string x and an integer L_K, which indicates the length (in bits) of output, as input and giving a bit string of length L_K as output, e.g. one of the dedicated hash-functions specified in ISO/IEC 10118-3
I2FE	a function that converts an integer into a field element
I2OS	a function that converts an integer into an octet string
I2P	a function that converts an integer into a point on the curve E
J,J_{DL},J_{EC}	functions for generating a password verification element from a password
k	the cofactor that is either the value $(q-1)/r$ in DL domain parameters or the value of $\#E/r$ in EC domain parameters
K	a function for deriving a key from a secret value and a key derivation parameter
K_1,K_2,...	secret keys established using a key establishment mechanism
L_K	the length (in bits) of an established secret key
m	an integer
M_i	an octet that is represented by values from 00 hex to FF hex
mod	binary operation, where $y = a \bmod b$ is defined to be the unique integer y satisfying $0 \leq y < b$ and $(a - y)$ is an integer multiple of b

n an integer

o_A, o_A', o_B, o_B' bit strings, which are used to specify a key confirmation process

OS2I a function that converts an octet string into an integer

p, p_i odd prime integers

P_1,P_2,... key derivation parameter octet strings

q the number of elements in the finite field $F(q)$. In the EC setting, q is either p or 2^m for some integer $m \geq 1$. In the DL setting, q is p

NOTE – this part of ISO/IEC 11770 treats only a prime field or a binary field in the EC setting and only a prime field in the DL setting, because these cases are widely used and their security properties have been well-explored.

r the order of the desired group, which is a prime dividing either $q - 1$ in the DL setting or $\#E$ in the EC setting

R,R_{1DL},R_{1EC}, R_{2DL},R_{2EC} functions for deriving a random element from a password

s_A, s_B Key token factors of entities A and B respectively, corresponding to key tokens w_A and w_B

NOTE – the key token factors should be generated at random from a selected range since this maximizes the difficulty of recovering the key token factor by collision-search methods. Methods of random number generation are specified in ISO/IEC 18031.

T a function for checking validity of a key token

V,V_A,V_B,V_{ADL},V_{AEC},V_{BDL},V_{BEC} functions for generating secret values

w_A, w_B key tokens or password-entangled key tokens of entities A and B respectively, corresponding to key token factors s_A and s_B; they are integers in the DL setting and points in the EC setting

$[x] \times Y$ multiplication operation in the EC setting that takes an integer x and a point Y on the curve E as input and produces a point Z on the curve E, where $Z = [x] \times Y = Y + Y + \ldots + Y$ adding x -1 times if x is positive. The operation satisfies $[0] \times Y = 0_E$ (the point at infinity), and $[-x] \times Y = [x] \times (-Y)$.

z a secret value used to derive the keys; it is an integer in the DL setting and a point in the EC setting

$\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ an element of $F(s^m)$ where s is either p or 2, and β_i is an integer satisfying $0 \leq \beta_i \leq s$ -1

π a password-based octet string which is generally derived from a password or a hashed password, identifiers for one or more entities, an identifier of a communication session if more than one session might execute concurrently, and optionally includes a salt value and/or other data

NOTE – It is required to include one or more the entity identifiers and a unique session identifier into the value of π, in order to avoid that a key establishment mechanism might be vulnerable to an unknown key-share attack addressed in [TC05].

$\#E$ the number of points on the elliptic curve E

|| concatenation operator, defined on octet strings

0_E the point at infinity on the elliptic curve E

5 Requirements

It is assumed that the entities are aware of each other's claimed identities. This may be achieved by the inclusion of identifiers in information exchanged between the two entities, or it may be apparent from the context of use of the mechanism.

It is assumed that the entities are aware of a common set of domain parameters, which are used to compute a variety of functions in the key establishment mechanism. Each mechanism can be used with one of two different sets of domain parameters, depending on whether the mechanism operates over the multiplicative group of values in $F(q)$ or over the additive group of elements in an elliptic curve defined over $F(q)$. In the first case the mechanism is said to operate in the DL (for "discrete logarithm") setting, and in the second case the mechanism is said to operate in the EC (for "elliptic curve") setting.

NOTE – It is fundamentally important to the correct operation of the mechanisms that any domain parameters are held correctly by each participant. Use by any party of accidentally or deliberately corrupted domain parameters can result in compromise of the mechanisms, which might allow an unauthorised third party to discover an established secret key.

The two sets of domain parameters are as follows.

A set of DL domain parameters consists of:

$F(q)$ – a specific representation of the finite field on q elements.

q – the number of elements in $F(q)$, which is an odd prime integer.

r – the order of the desired group of elements from the finite field, which is a prime divisor of q -1.

g – an element of multiplicative order r in $F(q)$ (g is called the generator of a subgroup of r elements in $F(q)$).

g_{q-1} – an element of multiplicative order q -1 in $F(q)$.

NOTE – a method of generating g_{q-1} can be found in Chapter 4 of [MvV96] and [Ka86].

k – the value $(q-1)/r$, also called the cofactor, satisfying $k = 2p_1p_2\ldots p_t$, for primes $p_i > r$, $i = 1, 2, \ldots, t$. Optionally, $t = 0$.

A set of EC domain parameters consists of:

$F(q)$ – a specific representation of the finite field on q elements.

q – the number of elements in $F(q)$, which is

— p, an odd prime integer, or

— 2^m for some positive integer $m \geq 1$.

a_1, a_2 – two elliptic curve coefficients, elements of $F(q)$, that define an elliptic curve E.

E – an elliptic curve defined by two elliptic curve coefficients, a_1 and a_2. It is defined by one of the following two equations

— $Y^2 = X^3 + a_1X + a_2$ over the field $F(p)$,

— $Y^2 + XY = X^3 + a_1X^2 + a_2$ over the field $F(2^m)$,

together with an extra point 0_E referred to as the point of infinity.

$\#E$ - the number of points on E.

r – the order of the desired group, which is a prime integer dividing $\#E$.

G – a curve point of order r (G is called the generator of a subgroup of r points on E).

k – the value $\#E/r$, also called the cofactor, satisfying $k = 2^n p_1 p_2 \ldots p_t$, for $n = \{0, 1, 2\}$ and primes $p_i > r$, $i = 1, 2, \ldots, t$. Optionally, $t = 0$.

When entities make use of a specified mechanism in the EC setting, it is assumed that the entities are aware of the form of the point representation, i.e., a point is represented in either compressed, uncompressed or hybrid form. The specifications in the point representation refer to ISO/IEC 18033-2.

In the mechanism specification of this part of ISO/IEC 11770, the method of random number generation refers to ISO/IEC 18031 and the method of prime number generation refers to ISO/IEC 18032.

It is also assumed that the entities are aware of a common hash-function H, e.g. one of the dedicated hash-functions specified in ISO/IEC 10118-3.

6 Password-authenticated key agreement

This clause specifies three password-authenticated key agreement mechanisms. The first mechanism, specified in clause 6.1, is a balanced password-authenticated key agreement mechanism, which requires the two entities to share a weak secret. The second and third mechanisms, specified in clauses 6.2 and 6.3 respectively, are augmented password-authenticated key agreement mechanisms, which require one of the two entities to possess verification data for a weak secret known to the other entity.

All three password-authenticated key agreement mechanisms have the following initialisation process and key establishment process.

Initialisation 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 may be publicly known. The two entities also agree to use either a shared password-based weak secret which is known only to them, or shared password-based information that means one entity has a password-based weak secret and the other entity has the corresponding password verification data.

Key establishment process:

1) *Generate and exchange key tokens.* The two entities involved each randomly choose one or more key token factors associated with the domain parameters, create the corresponding key tokens, which may be associated with the password or password verification data (a key token associated with the password or password verification data is called a password-entangled key token), and then make the key tokens available to the other entity.

2) *Check validity of key tokens.* Depending on the operations for producing key tokens in Step 1, the two entities involved each choose an appropriate method to validate the received key tokens based on the domain parameters. If any validation fails, output “invalid” and stop.

3) *Derive shared secret keys.* The two entities involved each apply certain secret value derivation functions to their own key token factor, the other entity's key tokens and/or shared password or password verification data 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.

4) *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 Mechanism 1 but mandatory in Mechanisms 2 and 3.

6.1 Key Agreement Mechanism 1

This key agreement mechanism is designed to achieve balanced password-authenticated key agreement, which establishes one or more shared secret keys between entities *A* and *B* with joint key control and prior sharing of a password-based octet string π. This mechanism provides mutual implicit key authentication and, optionally, mutual explicit key confirmation.

This mechanism works in both the DL and EC settings.

NOTE – This mechanism is based on the work of [Jab96] and the mechanism called {DL,EC}BPKAS-SPEKE in [IEEEP1363.2].

6.1.1 Prior shared parameters

The key agreement between two entities *A* and *B* takes place in an environment where the two entities share the following parameters:

— A shared password-based octet string π

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

— A random element derivation function, *R*

— A key token generation function, *D*

— A key token check function, *T*

— A secret value derivation function, *V*

— A key derivation function, *K*

— A Boolean value, *b*, which indicates whether cofactor multiplication is desired. If *b* = 1, cofactor multiplication is desired; otherwise it is not

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

— The length of a shared secret key, L_K

NOTE – Cofactor multiplication is used to map a received key token into a valid group element, i.e. an element in a selected subgroup of order *r*. *b* = 0 is only used in those mechanisms in which it is guaranteed that a received key token is a valid group element. More detailed discussion on cofactor multiplication can be found in [ISO/IEC 15946-3:2002].

6.1.2 Functions

6.1.2.1 Random element derivation function *R*

The random element derivation function *R* operates on an octet string *x* as input and produces a selected group element written $R(x)$ as output. Key Agreement Mechanism 1 can be used with any one of the following four *R* functions, R_{1DL}, R_{1EC}, R_{2DL} and R_{2EC}:

— R_{1DL} 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 *k* and *q*) and an octet string input *x*, R_{1DL} is defined as

$$R_{1DL}(x) = (BS2I(H(x)))^k \bmod q.$$

— R_{1EC} 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 *k*) and an octet string input *x*, R_{1EC} is defined as

$$R_{1EC}(x) = [k] \times I2P(BS2I(H(x))).$$

— R_{2DL} 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 q), two random elements in a subgroup of order r in $F(q)$, g_a and g_b, and an octet string input x, R_{2DL} is defined as

$$R_{2DL}(x) = g_a * g_b^{BS2I(H(x))} \bmod q.$$

— R_{2EC} 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, two random elements of a subgroup of order r on E, G_a and G_b, and an octet string input x, R_{2EC} is defined as

$$R_{2EC}(x) = G_a + [BS2I(H(x))] \times G_b.$$

Functions *BS2I* (Bit String to Integer conversion) and *I2P* (Integer to Point conversion) are described in Annex A.

NOTE 1 – The four choices for the function R allow for different performance characteristics and different security assumptions. Regarding performance, R_2 permits use where $k >> r$, but when using a small cofactor k, R_1 is faster than R_2.

NOTE 2 – It is recommended that, if the result of $R_{1DL}(x)$ or $R_{2DL}(x)$ is 1, or if the result of $R_{1EC}(x)$ or $R_{2EC}(x)$ is 0_E, output "invalid" and stop. Based on the randomness property of the hash-function H, this case happens with a negligible probability. However, there is no detected security weakness, because if the function R outputs the value 1 in the DL setting or the point 0_E in the EC setting without stopping, the protocol will abort when running the key token check function T.

6.1.2.2 Key token generation function *D*

The key token generation function D operates on an integer x and a group element y as input and produces another group element written $D(x, y)$ as output. Key Agreement Mechanism 1 can be used with either one of the following D functions, D_{DL} and D_{EC}:

— D_{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 q), and two inputs x from {1, …, r - 1} and an integer y the output of Function R, D_{DL} is defined as

$$D_{DL}(x, y) = y^x \bmod q.$$

— D_{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, and two inputs x from {1, …, r - 1} and a point Y the output of Function R, D_{EC} is defined as

$$D_{EC}(x, Y) = [x] \times Y.$$

6.1.2.3 Key token check function *T*

The key token check function T operates on a group element x as input and produces a Boolean value written $T(x)$ as output. Key Agreement Mechanism 1 can be used with either one of the following T functions, T_{DL} and T_{EC}:

— T_{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 q), and a data string x, T_{DL} is defined as follows:

  — If x does not represent an integer, $T_{DL}(x) = 0$.

  — If $x \leq 1$, $T_{DL}(x) = 0$.

  — If $x \geq q - 1$, $T_{DL}(x) = 0$.

— Else, $T_{DL}(x) = 1$.

— T_{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 0_E), a value $n \in \{0, 1, 2\}$, such that $k = 2^n p_1 p_2 \ldots p_t$ and a data string X, T_{EC} is defined as follows:

— If X does not represent a point on E, $T_{EC}(X) = 0$.

— If $[2^n] \times X = 0_E$, $T_{EC}(X) = 0$.

— Else, $T_{EC}(X) = 1$.

6.1.2.4 Secret value derivation function *V*

The secret value derivation function V operates on an integer x, a selected group element y and a Boolean value b as input and produces another group element written $V(x, y, b)$ as output. Key Agreement Mechanism 1 can choose one of the following V functions, V_{DL} and V_{EC}:

— V_{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 k and q), and three inputs, x from $\{1, \ldots, r - 1\}$, y from $\{2, \ldots, q - 2\}$ and b from $\{0, 1\}$, V_{DL} is defined as

$$V_{DL}(x, y, b) = y^{x*k^b} \bmod q.$$

— V_{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 k), and three inputs, x from $\{1, \ldots, r - 1\}$, a point Y ($\neq 0_E$) on the curve E and b from $\{0, 1\}$, V_{EC} is defined as

$$V_{EC}(x, Y, b) = [k^b * x] \times Y.$$

6.1.2.5 Key derivation function *K*

The key derivation function K operates on an octet string x , a length (in bits) L_K of the output of function K, and a key derivation parameter octet string P from $\{P_1, P_2, \ldots\}$ as input, and produces a bit string written $K(x, P, L_K)$ as output. Key Agreement Mechanism 1 makes use of a one-way function as Function K, i.e., given x, P and L_K as input, K is defined as

$$K(x, P, L_K) = h(x||P, L_K).$$

NOTE 1 – The output transformation for the hash-functions specified in ISO/IEC 10118-3 is the hash-code H with a given bit length. See ISO/IEC 10118-3 for the details.

NOTE 2 – The value of L_K is dependent on applications using the derived key. If the output of the key derivation function K is used as a key for a symmetric cipher, the value of L_K is the key length of a specific symmetric cipher mechanism.

6.1.3 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.

Key token construction (A1)

A performs the following steps:

— compute $g_1 = R(\pi)$ as a base of its key token,
— choose an integer s_A randomly from $\{1, \ldots, r - 1\}$ as its key token factor,
— compute $w_A = D(s_A, g_1)$ as the key token,
— make w_A available to B.

Key token construction (B1)

B performs the following steps:

— compute $g_1 = R(\pi)$ as a base of its key token,
— choose an integer s_B randomly from {1, …, r – 1} as its key token factor,
— compute $w_B = D(s_B, g_1)$ as the key token,
— make w_B available 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, carry on,
— compute $z = V(s_A, w_B, b)$ as a shared secret value,
— compute $K_i = K(GE2OS_X(z), P_i, L_K)$ for each key derivation parameter P_i as a shared secret key.

Shared secret key derivation (B2)

B performs the following steps:

— receive w_A from A,
— check validity of w_A using $T(w_A)$: if $T(w_A) = 0$, output "invalid" and stop; otherwise, carry on,
— compute $z = V(s_B, w_A, b)$ as a shared secret value,
— compute $K_i = K(GE2OS_X(z), P_i, L_K)$ for each key derivation parameter P_i as a shared secret key.

NOTE – No special ordering of steps A1 and B1 or A2 and B2 is specified, other than that logically required by the need to compute a value before using it, i.e., A2 and B2 must happen after A1 and B1.

Key confirmation (A3 and B3) (optional)

A performs the following steps (A3):

— compute $o_A = H(hex(03)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z)||GE2OS_X(g_1))$, and
— make o_A available to *B*.

B performs the following steps (B3):

— receive o_A from *A*,
— compute $o_A' = H(hex(03)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z)||GE2OS_X(g_1))$, and
— check if $o_A \neq o_A'$, output "invalid" and stop.

Key confirmation (B4 and A4) (optional)

B performs the following steps (B4):

— compute $o_B = H(hex(04)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z)||GE2OS_X(g_1))$, and
— make o_B available to *A*.

A performs the following steps (A4):

— receive o_B from *B*,
— compute $o_B' = H(hex(04)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z)||GE2OS_X(g_1))$, and
— check if $o_B \neq o_B'$, output "invalid" and stop.

NOTE – Entities *A* and *B* are free to choose A3 and B3, or B4 and A4. The only restriction is that B3 must happen after A3 and A4 must happen after B4.

Function $GE2OS_X$ (Group Element to Octet String conversion) is described in Annex A.

NOTE – 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.

6.2 Key Agreement Mechanism 2

This mechanism is designed to achieve augmented password-authenticated key agreement, which establishes one or more shared secret keys between entities *A* and *B* with joint key control. In the mechanism, *A* has a password-based octet string π and *B* has password verification data v corresponding to π. This mechanism provides unilateral explicit key authentication, and optionally mutual key confirmation.

This mechanism works in the DL setting.

NOTE 1 – In applications using augmented password-authenticated key agreement, *A* may play the role of a client and *B* may play the role of a server.

NOTE 2 – This mechanism is based on the work of [Wu02] and the mechanism called DLAPKAS-SRP6 in [IEEEP1363.2].

6.2.1 Prior shared parameters

The key agreement between two entities *A* and *B* takes place in an environment consisting of the following parameters:

- A set of DL domain parameters, including g_{q-1} and q, 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 key token generation function, *D*, used by *A*
- A password-entangled key token generation function, *C*, used by *B*
- 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* must agree to use the same P_i value
- An integer, *c*, defined as $c = (BS2I(H(I2OS(g_{q-1})||I2OS(q))) \bmod q$
- The length of a shared secret key, L_K

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

6.2.2 Functions

6.2.2.1 Password verification element derivation function *J*

The password verification element derivation function *J* operates on a password-based octet string π as input and produces an integer written $J(\pi)$ as output. Key Agreement Mechanism 2 can be used with the following *J* function:

- Given DL domain parameters (including g_{q-1} and q) and a password-based octet string input π, *J* is defined as,

$$J(\pi) = g_{q-1}^{BS2I(H(\pi))} \bmod q.$$

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

6.2.2.2 Key token generation function *D*

The key token generation function *D* operates on an integer *x* from $\{1, \ldots, q - 2\}$ as input, and produces an integer written $D(x)$ as output. Key Agreement Mechanism 2 can be used with the following *D* function:

- Given DL domain parameters (including g_{q-1} and q) and an input *x* from $\{1, \ldots, q - 2\}$, *D* is defined as,

$$D(x) = g_{q-1}^{x} \bmod q.$$

6.2.2.3 Password-entangled key token generation function *C*

The password-entangled key token generation function *C* operates on three inputs, the integer *c*, an integer *x* from {1, …, *q* - 2} and an output of the password verification element derivation function $v = J(\pi)$, and produces an integer written $C(x, v, c)$ as output. Key Agreement Mechanism 2 can be used with the following *C* function:

— Given the DL domain parameters (including g_{q-1} and *q*), and three inputs, the integer *c*, *x* from {1, …, *q* - 2} and the output of *J* function *v*, *C* is defined as,

$$C(x, v, c) = v * c + g_{q-1}{}^{x} \bmod q.$$

6.2.2.4 Secret value derivation functions V_A and V_B

1) The secret value derivation function V_A operates on six inputs, the integer *c*, a password-based octet string π, an integer x_A from {1, …, *q*-2}, an integer *v* that is the output of the password verification element derivation *J*, an integer y_A that is the output of the key token generation function *D*, and an integer y_B that is the output of the password-entangled key token generation function *C*, and produces an integer written $V_A(c, \pi, x_A, v, y_A, y_B)$ as output.

2) The secret value derivation function V_B operates on four inputs, an integer x_B from {1, …, *q*-2}, an integer *v* that is the output of the password verification element derivation *J*, an integer y_A that is the output of the key token generation function *D*, and an integer y_B that is the output of the password-entangled key token generation function *C*, and produces an integer written $V_B(x_B, v, y_A, y_B)$ as output.

3) V_A and V_B satisfy the condition $V_A(c, \pi, x_A, v, y_A, y_B) = V_B(x_B, v, y_A, y_B)$.

Key Agreement Mechanism 2 can be used with the following V_A and V_B functions:

1. Given the DL domain parameters (including gq -1 and q), the integer c, a password-based octet string π, an integer xA from {1, …, q - 2} , an output of J function v, an output of D function yA, and an output of C function yB, VA is defined in the following steps:

   — compute $u_1 = BS2I(H(\pi))$,

   — compute $u_2 = BS2I(H(I2OS(y_A)||I2OS(y_B)))$, and

   — compute $V_A(c, \pi, x_A, v, y_A, y_B) = (y_B - v * c)^{(x_A + u_1 * u_2)} \bmod q$.

2. Given the DL domain parameters (including g_{q-1} and *q*), an integer x_B from {1, …, *q* - 2}, an output of *J* function *v*, an output of *D* function y_A, and an output of *C* function y_B, V_B is defined in the following steps:

   — compute $u = BS2I(H(I2OS(y_A)||I2OS(y_B)))$, and

   — compute $V_B(x_B, v, y_A, y_B) = (y_A * v^{u})^{x_B} \bmod q$.

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

6.2.2.5 Key derivation function *K*

The key derivation function *K* is the same as defined in Clause 6.1.2.5.

6.2.3 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 A4 and B4 are optional.

Key token construction (A1)

A performs the following steps:

— choose an integer s_A randomly from {1, …, q – 2} as its key token factor,
— compute $w_A = D(s_A)$ as its key token, and
— make w_A available to *B*.

Password-entangled key token construction (B1)

B performs the following steps:

— receive w_A from *A*,
— check if $1 < w_A < q$ -1 holds, if not, output “invalid” and stop, otherwise carry on,
— choose an integer s_B randomly from {1, …, q – 2} as its key token factor,
— compute $w_B = C(s_B, v, c)$ as its password-entangled key token, and
— make w_B available to *A*.

Shared secret key derivation (A2)

A performs the following steps:

— receive w_B from *B*,
— check if $1 < w_B < q$ -1 holds, if not, output “invalid” and stop, otherwise carry on,
— compute $z = V_A(c, \pi, s_A, v, w_A, w_B)$ as an agreed secret value, and
— compute $K_i = K(I2OS(z), P_i, L_K)$ as a shared secret key for each key derivation parameter P_i.

Shared secret key derivation (B2)

B performs the following steps:

— compute $z = V_B(s_B, v, w_A, w_B)$ as an agreed secret value, and
— compute $K_i = K(I2OS(z), P_i, L_K)$ as a shared secret key for each key derivation parameter P_i.

Key confirmation (A3 and B3) (mandatory)

A performs the following steps:

— compute $o_A = H(hex(04)||I2OS(w_A)||I2OS(w_B)||I2OS(z)||I2OS(v))$, and
— make o_A available to *B*.

B performs the following steps:

— receive o_A from A,
— compute $o_A' = H(hex(04)||I2OS(w_A)||I2OS(w_B)||I2OS(z)||I2OS(v))$, and
— check if $o_A \neq o_A'$, output "invalid" and stop.

Key confirmation (B4 and A4) (optional)

B performs the following steps:

— compute $o_B = H(hex(03)||I2OS(w_A)||I2OS(w_B)||I2OS(z)||I2OS(v))$, and
— make o_B available to *A*.

A performs the following steps:

— receive o_B from B,
— compute $o_B' = H(hex(03)||I2OS(w_A)||I2OS(w_B)||I2OS(z)||I2OS(v))$, and
— check if $o_B \neq o_B'$, output "invalid" and stop.

Function *I2OS* (Integer to Octet String conversion) is described in Annex A.

NOTE – Entity *B* must verify the entity *A*'s proof of knowledge of the agreed key before revealing any information derived from the agreed key. Therefore, A3/B3 must be done before B4/A4, if the latter is performed.

6.3 Key Agreement Mechanism 3

This mechanism is designed to achieve augmented password-authenticated key agreement, which establishes one or more shared secret keys between entities *A* and *B* with joint key control. In the mechanism, *A* has a password-based octet string π and *B* has password verification data *v* corresponding to π. This mechanism provides unilateral explicit key authentication, and optionally mutual key confirmation.

This mechanism works in both the DL setting and the EC setting.

NOTE 1 – In applications using augmented password-authenticated key agreement, *A* may play the role of a client and *B* may play the role of a server.

NOTE 2 – This mechanism is based on the work of [Kw00] and [Kw03] and the mechanism called {DL, EC}APKAS-AMP in [IEEEP1363.2].

6.3.1 Prior shared parameters

The key agreement between two entities *A* and *B* takes place in an environment 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 key token generation function, *D*, used by *A*
- A password-entangled key token generation function, *C*, 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* must agree to use the same P_i value
- The length of a shared secret key, L_K

6.3.2 Functions

6.3.2.1 Password verification element derivation function *J*

The password verification element derivation function *J* operates on a password-based octet string π as input and produces a selected group element defined over $F(q)$ written $J(\pi)$ as output. Key Agreement Mechanism 3 can be used with either one of the following two *J* functions, J_{DL} *and* J_{EC}:

- 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* and *q*), and a password-based octet string π, J_{DL} is defined as,

$$J_{DL}(\pi) = g^{BS2I(H(\pi))} \bmod q.$$

- 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*), and a password-based octet string π, J_{EC} is defined as,

$$J_{EC}(\pi) = [BS2I(H(\pi))] \times G.$$

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

6.3.2.2 Key token generation function *D*

The key token generation function *D* operates on an integer *x* from {1, …, *r* - 1} as input, and produces a selected group element written *D*(*x*) as output. Key Agreement Mechanism 3 can be used with either one of the following two *D* functions, D_{DL} and D_{EC}:

— D_{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*), and an input *x* from {1, …, *r* - 1}, D_{DL} is defined as,

$$D_{DL}(x) = g^x \bmod q.$$

— D_{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*), and an input *x* from {1, …, *r* - 1}, D_{EC} is defined as,

$$D_{EC}(x) = [x] \times G.$$

6.3.2.3 Password-entangled key token generation function *C*

The password-entangled key token generation function *C* operates on three inputs, an integer *x* from {1, …, *r* - 1}, an output of *J* function *v* (or V), and an output of *D* function *y* (or *Y*), and produces a selected group element written *C*(*x*, *v*, *y*) as output. Key Agreement Mechanism 3 can be used with either one of the following *C* functions, C_{DL} and C_{EC}:

— 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*). Given the DL domain parameters (including *q*), and three inputs, *x* from {1, …, *r* - 1}, and *v* the output of Function *J*, and *y* the output of Function *D*, C_{DL} is defined as follows:

  — compute $e = BS2I(H(I2OS(1)||GE2OS_X(y)))$,

  — compute $C_{DL}(x, v, y) = (v * y^e)^x \bmod q$,

  — check if $C_{DL}(x, v, y)$ is 1 or *q* - 1, output "invalid" and stop; otherwise output $C_{DL}(x, v, y)$.

— 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, and three inputs, *x* from {1, …, *r* - 1} and *V* the output of Function *J* and *Y* the output of Function *D*, C_{EC} is defined as follows:

  — compute $e = BS2I(H(I2OS(1)||GE2OS_X(Y)))$,

  — compute $C_{EC}(x, V, Y) = [x] \times (V + [e] \times Y)$,

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

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.

6.3.2.4 Key token check function *T*

The key token check function *T* is the same as defined in Clause 6.1.2.3.

6.3.2.5 Secret value derivation functions V_A and V_B

1) The secret value derivation function V_A operates on four inputs, a password-based octet string π, an integer x_A from {1, …, r - 1}, an output of D function y_A (or Y_A), and an output of C function y_B (or Y_B), and produces a selected group element written $V_A(\pi, x_A, y_A, y_B)$ as output.

2) The secret value derivation function V_B operates on three inputs, an integer x_B from {1, …, r - 1}, an output of D function y_A (or Y_A) and an output of C function y_B (or Y_B), and produces a selected group element written $V_B(x_B, y_A, y_B)$ as output.

3) V_A and V_B satisfy the condition $V_A(\pi, x_A, y_A, y_B) = V_B(x_B, y_A, y_B)$.

Key Agreement Mechanism 3 can be used with either one of the following two V_A functions, V_{ADL} and V_{AEC}, and either one of the following two V_B functions, V_{BDL} and V_{BEC}:

1. 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), a password-based octet string π, an integer x_A from {1, …, r - 1}, an integer y_A from {2, …, q - 2}, and an integer y_B from {2, …, q - 2}, V_{ADL} is defined in the following steps:

   — compute $e = BS2I(H(I2OS(1)||GE2OS_X(y_A)))$,

   — compute $d = BS2I(H(I2OS(2)||GE2OS_X(y_A)||GE2OS_X(y_B)))$,

   — compute $u = (x_A + d)/(x_A * e + BS2I(H(\pi))) \bmod r$,

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

   — output $V_{ADL}(\pi, x_A, y_A, y_B)$.

2. 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 $F(q)$. Given the DL domain parameters (including g and q), an integer x_B from {1, …, r - 1}, an integer y_A from {2, …, q - 2}, and an integer y_B from {2, …, q - 2}, V_{BDL} is defined in the following steps:

   — compute $d = BS2I(H(I2OS(2)||GE2OS_X(y_A)||GE2OS_X(y_B)))$,

   — compute $V_{BDL}(x_B, y_A, y_B) = (y_A * g^{d})^{x_B} \bmod q$,

   — output $V_{BDL}(x_B, y_A, y_B)$.

3. 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), a password-based octet string π, an integer x_A from {1, …, r - 1}, a point Y_A ($\neq 0_E$) on E, and a point Y_B ($\neq 0_E$) on E, V_{AEC} is defined in the following steps:

   — compute $e = BS2I(H(I2OS(1)||GE2OS_X(Y_A)))$,

   — compute $d = BS2I(H(I2OS(2)||GE2OS_X(Y_A)||GE2OS_X(Y_B)))$,

   — compute $u = (x_A + d)/(x_A * e + BS2I(H(\pi))) \bmod r$,

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

   — output $V_{AEC}(\pi, x_A, Y_A, Y_B)$.

4. 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 G), an integer x_B from {1, …, r - 1}, a point Y_A ($\neq 0_E$) on E, and a point Y_B ($\neq 0_E$) on E, V_{BEC} is defined in the following steps:

   — compute $d = BS2I(H(I2OS(2)||GE2OS_X(Y_A)||GE2OS_X(Y_B)))$,

   — compute $V_{BEC}(x_B, Y_A, Y_B) = [x_B] \times (Y_A + [d] \times G)$,

   — output $V_{BEC}(x_B, Y_A, Y_B)$.

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.

6.3.2.6 Key derivation function *K*

The key derivation function K is the same as defined in Clause 6.1.2.5.

6.3.3 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 A4 and B4 are optional.

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, and
— make w_A available to *B*.

Password-entangled key token construction (B1)

B performs the following steps:

— receive w_A from *A*,
— check validity of w_A using $T(w_A)$: if $T(w_A) = 0$, output “invalid” and stop; otherwise carry on,
— choose an integer s_B randomly from {1, …, r – 1} as its key token factor,
— compute $w_B = C(s_B, v, w_A)$ as its password-entangled key token (if the output of function C is "invalid", go back to the above item to choose a different s_B value at random and try again), and
— make w_B available 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 carry on,
— compute $z = V_A(\pi, s_A, w_A, w_B)$ as an agreed secret value, and
— compute $K_i = K(GE2OS_X(z), P_i, L_K)$ as a shared secret key for each key derivation parameter P_i.

Shared secret key derivation (B2)

B performs the following steps:

— compute $z = V_B(s_B, w_A, w_B)$ as an agreed secret value, and
— compute $K_i = K(GE2OS_X(z), P_i, L_K)$ as a shared secret key for each key derivation parameter P_i.

Key confirmation (A3 and B3) (mandatory)

A performs the following steps (A3):

— compute $o_A = H(I2OS(4)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z))$, and
— make o_A available to *B*.

B performs the following steps (B3):

— receive o_A from A,
— compute $o_A' = H(I2OS(4)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z))$, and
— check if $o_A \neq o_A'$, output "invalid" and stop.

Key confirmation (B4 and A4) (optional)

B performs the following steps (B4):

— compute $o_B = H(I2OS(3)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z))$, and

— make o_B available to *A*.

A performs the following steps (A4):

— receive o_B from B,

— compute $o_B' = H(I2OS(3)||GE2OS_X(w_A)||GE2OS_X(w_B)||GE2OS_X(z))$, and

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

Function $GE2OS_X$ (Group Element to Octet String conversion) is 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 – Entity *B* must verify the entity *A*'s proof of knowledge of the agreed key before revealing any information derived from the agreed key. Therefore, A3/B3 must be done before B4/A4, if the latter is performed.

NOTE 3 – Based on the Pohlig-Hellman decomposition attack, the lowest one or two bits of *B*'s secret value s_B may be discernable by an attacker, when k is divisible by 2 or 4.

7 Password-authenticated key retrieval

This clause specifies a password-authenticated key retrieval mechanism. In the mechanism, one entity *A* has a weak secret derived from a password, and the other entity *B* has a strong secret associated with *A*'s weak secret. Using their respective secrets, the two entities negotiate a secret key, which is retrievable by *A* but not necessarily derivable by *B*.

The result of the process is that *A* retrieves the value of a secret key that is derived from both its own weak secret and *B*'s strong secret. *B* does not need to know either *A*'s secret or the resulting secret key. *B*'s secret is associated with the *A*'s secret, but does not (in itself) contain sufficient information to permit either *A*'s secret or the established secret key to be determined, even with a brute-force attack.

NOTE – In applications using password-authenticated key retrieval, *A* may play role of a client and *B* may play role of a server.

A password-authenticated key retrieval operation has the following initialisation process and key retrieval process.

Initialisation process: Entities *A* and *B* agree to use a set of valid domain parameters and a set of functions, both of which may be publicly known. *A* establishes a password-based weak secret and *B* establishes a strong secret associated with *A*'s weak secret.

Key establishment process:

1) *Generate and exchange key tokens.* Entity *A* selects a key token factor, constructs its password-entangled key token, and makes the key token available to entity *B*. After receiving *A*'s password-entangled key token, *B* constructs its key token, and makes the key token available to *A*.

2) *Check validity of key tokens.* (*Optional*) Depending on the operations for producing key tokens, entities *A* and *B* each choose an appropriate method to validate the received key contributions and the domain parameters. If any validation fails, output "invalid" and stop.

3) *Derive a static secret key. A* applies cryptographic operations to its own key token factor and entity *B*'s key token to produce a secret value, and further applies a key derivation function to the secret value and one or more key derivation parameters to produce one or more secret keys.

7.1 Key Retrieval Mechanism 1

This mechanism is designed to achieve password-authenticated key retrieval. It uses a password to derive the generator for a modified form of Diffie-Hellman key agreement. Entity *B* determines the key to be distributed to entity *A*.

This mechanism works in both the DL setting and the EC setting.

NOTE – This mechanism is based on the work of [FK00] and the mechansim called {DL,EC}PKRS-1 in [IEEEP1363.2].

7.1.1 Prior shared parameters

The key retrieval operation involving two entities *A* and *B* takes place in an environment 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 known only to *A*
- A secret integer s_B from {1, …, r – 1} used for *B*'s key token factor and known only to *B*
- A random element derivation function, *R*, used by *A*
- A key token generation function, *D*, used by both *A* and *B*
- A key token check function, *T*
- A secret value derivation function, *V*, used by *A*
- A key derivation function, *K,* used by *A*
- One or more key derivation parameter octet strings {P_1, P_2, ...}
- The length of a result secret key, L_K

7.1.2 Functions

7.1.2.1 Random element derivation function *R*

This function is R_{1DL} or R_{1EC} as defined in Clause 6.1.2.1.

7.1.2.2 Key token generation function *D*

The key token generation function *D* is the same as specified in Clause 6.1.2.2.

7.1.2.3 Key token check function *T*

This function is the same as defined in Clause 6.1.2.3.

7.1.2.4 Secret value derivation function *V*

The secret value derivation function *V* operates on an integer *x* and a selected group element *y* as input and produces another group element written *V*(*x*, *y*) as output. Key Retrieval Mechanism 1 can be used with either one of the following two *V* functions, V_{DL} and V_{EC}:

— V_{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 r and q), and two inputs, x from {1, …, r - 1} and y from {2, …, q - 2}, V_{DL} is defined as

$$V_{DL}(x, y) = y^{x^{-1} \bmod r} \bmod q.$$

— V_{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 r), and two inputs, x from {1, …, r - 1} and a point Y ($\neq 0_E$) on E, V_{EC} is defined as

$$V_{EC}(x, Y) = [x^{-1} \bmod r] \times Y.$$

7.1.2.5 Key derivation function *K*

This function is the same as defined in Clause 6.1.2.5.

7.1.3 Key retrieval operation

This mechanism involves *A* performing a sequence of up to two steps, numbered A1and A2, and *B* performing one step, numbered B1.

Key token construction (A1)

A performs the following steps:

— Compute $g_1 = R(\pi)$ as a base element of its key token,
— choose an integer s_A randomly from {1, …, r – 1} as its key token factor,
— computes $w_A = D(s_A, g_1)$ as its key token, and
— make w_A available to *B*.

Key token construction (B1)

B performs the following steps:

— receive w_A from *A*,
— check validity of w_A using $T(w_A)$: if $T(w_A) = 0$, output “invalid” and stop; otherwise carry on,
— computes $w_B = D(s_B, w_A)$ as its key token, and
— make w_B available to *A*.

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 carry on,
— compute $z = V(s_A, w_B)$ as its hardened secret,
— compute $K_i = K(GE2OS_X(z), P_i, L_K)$ for each key derivation parameter octet string P_i in {P_1, P_2, ...} as a secret key.

Function $GE2OS_X$ (Group Element to Octet String conversion) is 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 – Based on the Pohlig-Hellman decomposition attack, the lowest one or two bits of *B*'s secret value s_B may be discernable by an attacker, when k is divisible by 2 or 4.

Annex A
(normative)

Functions for Data Type Conversion

This annex specifies a few data type conversion functions which are used as part of key establishment mechanisms in this part of ISO/IEC 11770.

A.1 *I2OS & OS2I*

This section specifies Functions *I2OS* (Integer to Octet String conversion) and *OS2I* (Octet String to Integer conversion).

Function *I2OS* takes as input a non-negative integer x, and produces the unique octet string $M_{l-1}M_{l-2} \ldots M_0$ of length l as output, where $l = \lceil \log_{256}(x + 1) \rceil$ is the length in octets of x. *I2OS* is defined as follows:

1. Write x in its unique l-digit base 256 representation:

$$x = x_{l-1}\, 256^{l-1} + x_{l-2}\, 256^{l-2} + \ldots + x_1\, 256 + x_0,$$

   where $0 \le x_i < 256$.

2. Let the octet M_i have the value x_i for $0 \le i \le l-1$.

3. Output the octet string $M_{l-1}\, M_{l-2} \ldots M_0$.

For example, *I2OS*(10945) = 2A C1.

Function *OS2I* takes an octet string $M_{l-1}\, M_{l-2} \ldots M_0$ as input and produces a non-negative integer y as output. It is defined as follows:

1. Let integer y_i have the value of the octet M_i for $0 \le i \le l-1$.

2. Compute the integer $y = y_{l-1}\, 256^{l-1} + y_{l-2}\, 256^{l-2} + \ldots + y_1\, 256 + y_0$.

3. Output y.

For example, *OS2I*(2A C1) = 10945.

Note that the octet string of length zero (the empty octet string) is converted to the integer 0 and vice versa.

A.2 *BS2I*

This section specifies Function *BS2I* (Bit String to Integer conversion).

Function *BS2I* takes a bit string $b_{l-1}\, b_{l-2} \ldots b_0$ as input and produces a non-negative integer as output. It is defined as follows:

1. Let integer y_i have the value of the bit b_i for $0 \le i \le l-1$.

2. Compute the integer $y = y_{l-1}\, 2^{l-1} + y_{l-2}\, 2^{l-2} + \ldots + y_1\, 2 + y_0$.

3. Output y.

For example, if l = 19, *BS2I*(000 0010 1010 1100 0001) = 10945.

Note that the bit string of length zero (the empty bit string) is converted to the integer 0.

A.3 *FE2I & I2FE*

This section specifies Functions *FE2I* (Field Element to Integer conversion) and *I2FE* (Integer to Field Element conversion).

Let an element of a finite field $F(s^m)$ (where s is either p or 2) be represented by $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ where β_i is an integer satisfying $0 \le \beta_i \le s-1$.

Function *FE2I* takes an element $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ as input and produces a non-negative integer as output. It is defined as follows:

1. Let integer y_i have the value of β_i for $0 \le i \le m-1$.
2. Compute the integer $y = y_{m-1}\, s^{m-1} + y_{m-2}\, s^{m-2} + \ldots + y_1\, s + y_0$.
3. Output y.

Function *I2FE* takes a non-negative integer x as input and produces an element $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ as output. It is defined as follows:

1. Write x in its unique m-digit base s representation:

   $$x = x_{m-1}\, s^{m-1} + x_{m-2}\, s^{m-2} + \ldots + x_1\, s + x_0,$$

   where $0 \le x_i < s$ (note that one or more leading digits will be zero if $x < s^{m-1}$).
2. Let β_i have the value x_i for $0 \le i \le m-1$.
3. Output $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$.

A.4 *FE2OS*

This section specifies Function *FE2OS* (Field Element to Octet String conversion).

Function *FE2OS* takes an element $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ as input and produces an octet string y as output. It is defined as follows:

1. Convert $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ in to an integer x by using *FE2I*.
2. Convert x into an octet string y by using *I2OS*.

A.5 *GE2OS$_X$*

This section specifies Function *GE2OS$_X$* (Group Element to Octet String conversion).

Function *GE2OS$_X$* takes a group element as input and produces an octet string as output. It is defined as follows:

In the DL setting, group elements are elements in $F(q)$. Let u be a group element. The output of $GE2OS_X(u)$ is computed as follows:

1. Represent u as a field element.
2. Convert the result of 1) into an octet string using *FE2OS*.
3. Output the result of 2).

In the EC setting, group elements are points on the elliptic curve E. Let $Q = (x_Q, y_Q)$ be a point on E, where x_Q is the x-coordinate of Q and y_Q is the y-coordinate of Q; both x_Q and y_Q are elements in $F(q)$. For the purpose of specifying mechanisms in this part of ISO/IEC 11770, the function $GE2OS_X(Q)$ converts the x-coordinate of Q to an octet string and ignore the y-coordinate of Q. The output of $GE2OS_X(Q)$ is computed as follows:

1. Represent x_Q as a field element.
2. Convert the result of 1) into an octet string using *FE2OS*.
3. Output the result of 2).

NOTE – This conversion does not define a 1-1 mapping. For example, this conversion will associate the elliptic curve points Q and $-Q$ with the same octet string.

A.6 *I2P*

This section specifies Function *I2P* (Integer to Point conversion).

Given a set of EC domain parameters (E, q, p, m, r, k, a_1, a_2), Function *I2P* operates on an integer u as input, and produces a point T on the curve E over $F(q)$ as output, which is specified as $T = I2P(u)$. In the following specification, the operations of addition and multiplication between finite field elements follow the specification in ISO/IEC 15946-1.

1. Set $v = BS2I(H(I2OS(u)))$ mod q.
   - If $v = 0$, output "invalid" and stop.
2. Set $\lambda = u$ mod 2.
3. If q is prime ($q = p$) and the curve E is $Y^2 = X^3 + a_1X + a_2$ defined over $F(p)$, compute the point T in the following way:
   (a) Set $x = v$.
   (b) Compute the field element $\alpha = x^3 + a_1x + a_2$ mod p.
      - If $\alpha = 0$, output "invalid" and stop.
   (c) Find a square root β of α modulo p (i.e., an integer β with $0 < \beta < p$ such that $\beta^2 = \alpha$ mod p) or determine that no such square roots exist.
      - To determine the existence of the square root, compute $\delta = \alpha^{(p-1)/2}$ mod p. If $\delta = 1$, β exists, otherwise β does not exist.
      - If $\delta \neq 1$, compute $u = u + 1$ mod p and go to Step 1.
      - If $\delta = 1$, find β.