INTERNATIONAL STANDARD

**ISO/IEC 11770-4**

Second edition
2017-11

# 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 11770-4:2017(E)



ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

# Contents

Page

# Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents 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 shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

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

For an explanation on the 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 the following URL: www.iso.org/iso/foreword.html.

This document was prepared by ISO/IEC JTC 1, *Information technology*, SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 11770-4:2006), which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC 11770-4:2006/Cor1:2009.

This edition includes the following significant changes with respect to the previous edition:

— revision of the Balanced Key Agreement Mechanism 1 (BKAM1) to address the attacks reported in Reference [6];

— addition of a new Balanced Key Agreement Mechanism 2 (BKAM2) based on the J-PAKE scheme of Reference [5];

— addition of a new Augmented Key Agreement Mechanism 3 (AKAM3) based on the AugPAKE scheme of Reference [23].

A list of all parts in the ISO/IEC 11770 series can be found on the ISO website.



 

# Introduction

The mechanisms specified in this document are designed to achieve one of the following three goals.

a) **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.

b) **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 1 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*. A typical application scenario would involve use between a client (*A*) and a server (*B*).

c) **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 2 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. Such a mechanism is appropriate for use between a client (*A*) and a server (*B*).

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

ISO and IEC take no position concerning the evidence, validity and scope of these patent rights. The holders of these patent rights have assured ISO and IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statements of the holders of these patent rights are registered with ISO and IEC. Information may be obtained from:

National Institute of Advanced Industrial Science and Technology

1–1–1 Umezono

Tsukuba, Ibaraki

305–8560 Japan

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

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

# Information technology — Security techniques — Key management —

## Part 4: Mechanisms based on weak secrets



## 1 Scope

This document 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 offline brute-force attacks associated with the weak secret. This document is not applicable to the following aspects of key management:

— life-cycle 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.

## 2 Normative reference

There are no normative references in this document.

## 3 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

**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 1 to entry: Computational feasibility depends on the specific security requirements and environment. Refer to ISO/IEC 10118-1:2016, Annex C.

[SOURCE: ISO/IEC 10118-1:2016, 3.1]

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

Note 1 to entry: 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 1 to entry: 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.

[SOURCE: ISO/IEC 9796-3:2006, 3.2]

**3.7**
**elliptic curve**
cubic curve $E$ without a singular point

Note 1 to entry: The set of points $E$ together with an appropriately defined operation for a field that includes all coefficients of the equation describing $E$ is called the definition field of $E$. In ISO/IEC 15946-1, only finite fields $F$ are dealt with as the definition field. When it is necessary to describe the definition field $F$ of $E$ explicitly, the curve is denoted as $E/F$.

Note 2 to entry: The form of a cubic curve equation used to define an elliptic curve varies depending on the field. The general form of an appropriate cubic equation for all possible finite fields is defined in ISO/IEC 15946-1:2016, 6.1.

[SOURCE: ISO/IEC 15946-1:2016, 3.3, modified]

**3.8**
**explicit key authentication**
<from entity $A$ to entity $B$> assurance for entity $B$ that entity $A$ is the only other entity that is in possession of the correct key

Note 1 to entry: Implicit key authentication from entity $A$ to entity $B$ and key confirmation from entity $A$ to entity $B$ together imply explicit key authentication from entity $A$ to entity $B$.

[SOURCE: ISO/IEC 11770-3:2015, 3.12, modified]

**3.9**
**hash-function**
function which maps strings of bits of variable (but usually upper bounded) length to fixed-length strings of bits, satisfying the following two properties:

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

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

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment. Refer to ISO/IEC 10118-1:2016, Annex C.

[SOURCE: ISO/IEC 10118-1:2016, 3.4]

**3.10**
**hashed password**
result of applying a hash-function to a password

**3.11**
**implicit key authentication**
<from entity *A* to entity *B*> assurance for entity *B* that entity *A* is the only other entity that can possibly be in possession of the correct key

[SOURCE: ISO/IEC 11770-3:2015, 3.16, modified]

**3.12**
**key**
sequence of symbols that controls the operation of a cryptographic transformation (e.g. encryption, decryption, cryptographic check function computation, signature calculation, or signature verification)

[SOURCE: ISO/IEC 11770-3:2015, 3.17]

**3.13**
**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

Note 1 to entry: By predetermine, it is meant that neither entity *A* nor entity *B* can, in a computationally efficient way, choose a smaller key space and force the computed key in the protocol to fall into that key space.

[SOURCE: ISO/IEC 11770-3:2015, 3.18]

**3.14**
**key confirmation**
<from entity *A* to entity *B*> assurance for entity *B* that entity *A* is in possession of the correct key

[SOURCE: ISO/IEC 11770-3:2015, 3.20, modified]

**3.15**
**key control**
ability to choose the key or the parameters used in the key computation

[SOURCE: ISO/IEC 11770-3:2015, 3.21]

**3.16**
**key derivation function**
function which takes as input a number of parameters, at least one of which shall be secret, and which gives as output keys appropriate for the intended algorithm(s) and applications

**3.17**
**key establishment**
process of making available a shared secret key to one or more entities

Note 1 to entry: Key establishment includes key agreement, key transport and key retrieval.

**3.18**
**key management**
administration and use of generation, registration, certification, deregistration, distribution, installation, storage, archiving, revocation, derivation and destruction of keying material in accordance with a security policy

[SOURCE: ISO/IEC 11770-1:2010, 2.28]

**3.19**
**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

**3.20**
**key token**
key establishment message sent from one entity to another entity during the execution of a key establishment mechanism

**3.21**
**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

**3.22**
**key token factor**
value that is kept secret and that is used, possibly in conjunction with a weak secret, to create a key token

**3.23**
**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

**3.24**
**message authentication code algorithm**
algorithm for computing a function which maps strings of bits and a secret key to fixed-length strings of bits, satisfying the following two properties:

— for any key and any input string, the function can be computed efficiently;

— for any fixed key, and given no prior knowledge of the key, it is computationally infeasible to compute the function value on any new input string, even given knowledge of a set of input strings and corresponding function values, where the value of the *i*th input string may have been chosen after observing the value of the first *i*-1 function values (for integers $i > 1$)

[SOURCE: ISO/IEC 9797-1:2011, 3.10, modified]

**3.25**
**mutual key authentication**
assurance for two entities that only the other entity is in possession of the correct key

**3.26**
**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 an input which maps to a given output

[SOURCE: ISO/IEC 11770-3:2015, 3.30]

**3.27**
**password**
secret word, phrase, number, or character sequence used for entity authentication, which is a memorized weak secret



 

**3.28**
**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

**3.29**
**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*

**3.30**
**password-entangled key token**
key token which is derived from both a weak secret and a key token factor

**3.31**
**password verification data**
data that is used to verify an entity's knowledge of a specific password

**3.32**
**random element derivation function**
function that utilizes a password and other parameters as input and outputs a random element

**3.33**
**salt**
random variable incorporated as secondary input to a one-way or encryption function that is used to derive password verification data

**3.34**
**secret**
value known only to authorized entities

**3.35**
**secret key**
key used with symmetric cryptographic techniques by a specified set of entities

[SOURCE: ISO/IEC 11770-3:2015, 3.36]

**3.36**
**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

**3.37**
**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 1 to entry: This may, for example, be achieved by randomly choosing the secret from a sufficiently large set of possible values under uniform distribution.

**3.38**
**weak secret**
secret that can be conveniently memorized by a human being

Note 1 to entry: Typically, this means that the entropy of the secret is limited, so that an exhaustive search for the secret (or, a dictionary attack) may be feasible, given knowledge that would enable a correct guess for the secret to be distinguished from an incorrect guess.

## 4 Symbols and abbreviated terms

| | |
|---|---|
| $A$, $B$ | distinguishing identities of entities represented as octet strings |
| $a_1$, $a_2$ | elliptic curve coefficients |
| BS2I | a function that converts a bit string into an integer (described in Annex A) |
| $b$, $b_i$ | bits (i.e. either 0 or 1) |
| $C$, $C_{DL}$, $C_{EC}$ | functions for generating a key token based on a password and a key token factor |
| $c$ | an integer satisfying $1 \leq c \leq q - 1$ |
| $D$, $D_{DL}$, $D_{EC}$ | 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 $a_2$. For the purpose of this document, an elliptic curve is not only the set of points on the curve, but also a group operation defined on these points as specified in ISO/IEC 15946-1[13]. |
| FE2I | a function that converts a field element into an integer (described in Annex A) |
| FE2OS | a function that converts a field element into an octet string (described in Annex A) |
| $F(q)$ | the finite field with $q$ elements |
| $G$,$G_a$,$G_b$ | points of order $r$ on $E$ over $F(q)$ |
| $GE2OS_X$ | 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 (described in Annex A) |
| $g$, $g_1$, $g_a$, $g_b$ | elements of multiplicative order $r$ in $F(q)$ |
| $g_{q\text{-}1}$ | an element of multiplicative order $q$ - 1 in $F(q)$ |
| $H$ | a collision-resistant hash-function taking an octet string as input and giving a bit string as output, e.g. based on one of the dedicated hash-functions specified in ISO/IEC 10118-3[11] |
| $h(x, L_K)$ | a collision-resistant hash-function taking an octet string x and an integer $L_K$ as input and giving a bit string of length $L_K$ (in bits) as output, e.g. based on one of the dedicated hash-functions specified in ISO/IEC 10118-3[11] |
| I2FE | a function that converts an integer into a field element (described in Annex A) |
| I2OS | a function that converts an integer into an octet string (described in Annex A) |
| I2P | a function that converts an integer into a point on the curve $E$ (described in Annex A) |
| $J$, $J_{DL}$, $J_{EC}$ | functions for generating a password verification element from a password |
| $K$ | a function for deriving a key from a secret value and a key derivation parameter |
| $KC\_1\_U$ | an octet string of the constant value “KC_1_U” that literally means unilateral key confirmation |
| $K_1$, $K_2$, ... | secret keys established using a key establishment mechanism |
| $k$ | the cofactor that is either the value $(q - 1)/r$ in DL domain parameters or the value of $\#E/r$ in EC domain parameters |



 

| | |
|---|---|
| $L_K$ | the length (in bits) of an established secret key |
| *MAX* | a function that takes two integers as input and outputs the integer with a larger value; if the two integers are identical, it outputs an error message |
| *MIN* | a function that takes two integers as input and outputs the integer with a smaller value; if the two integers are identical, it outputs an error message |
| $M_i$ | an octet that is represented by values from 00 hex to FF hex |
| $m$ | an integer |
| *mac*($k$, $m$) | a message authentication code (MAC) function taking a key $k$ and a variable-length message $m$ as input and giving a fixed-length output, e.g., by using one of the MAC algorithms specified in ISO/IEC 9797-2[10] |
| 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 |
| null | an empty octet string with the byte length zero |
| OS2I | a function that converts an octet string into an integer (described in Annex A) |
| $o_A$, $o_A'$, $o_B$, $o_B'$ | bit strings, which are used to specify a key confirmation process |
| $P_1$, $P_2$, ... | key derivation parameter octet strings |
| $p$, $p_i$ | odd prime integers |
| $q$ | 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 $p$.<br>NOTE 1 This document 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. |
| $R$, $R_{1DL}$, $R_{1EC}$, $R_{2DL}$, $R_{2EC}$ | functions for deriving a random element from a password |
| $r$ | the order of the desired group, which is a prime dividing either $q - 1$ in the DL setting or $\#E$ in the EC setting |
| $s_A$, $s_B$ | key token factors of entities $A$ and $B$, respectively, corresponding to key tokens $w_A$ and $w_B$. 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[14]. |
| $T$ | 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 |
| $z$ | a secret value used to derive the keys; it is an integer in the DL setting and a point in the EC setting |

| | |
|---|---|
| $\pi$ | a password-based octet string which is generally derived from the following data items: a) a password or a hashed password, b) identities for one or more entities, c) an identity of a communication session if more than one session may execute concurrently, and d) a salt value and/or other data. All items except the first are optional.<br>NOTE 2 As addressed in Reference [26], including entity and session identities in the computation of $\pi$ can prevent an unknown key-share attack. As further addressed in Reference [6], putting these identities into the session key computation function rather than in the computation of $\pi$ can also avoid a number of attacks, including the unknown key-share attack. |
| $\#E$ | the number of points on the elliptic curve, $E$ |
| $[x] \times Y$ | 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 $[-x] \times Y = [x] \times (-Y)$. |
| $\{\beta_{m-1}, \beta_{m-2}, \ldots, \beta_0\}$ | an element of $F(s^m)$ where $s$ is either $p$ or 2, and $\beta_i$ is an integer satisfying $0 \leq \beta_i \leq s - 1$ |
| $\|\|$ | $X \|\| Y$ denotes the result of the concatenation of octet strings $X$ and $Y$ in the order specified. In cases where the result of concatenating two or more octet strings is input to a cryptographic function as part of one of the mechanisms specified in this document, this result shall be composed so that it can be uniquely resolved into its constituent octet strings, i.e. so that there is no possibility of ambiguity in interpretation. This latter property could be achieved in a variety of different ways, depending on the application. For example, it could be guaranteed by a) fixing the length of each of the octet strings throughout the domain of use of the mechanism, or b) encoding the sequence of concatenated octet strings using a method that guarantees unique decoding, e.g. using the distinguished encoding rules defined in ISO/IEC 8825-1[9]. |
| $0_E$ | 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 identities in information exchanged between the two entities, or it may be apparent from the context of use of the mechanism.

It is assumed that if the entities are engaged in several sessions in parallel, the entities are aware of a unique session identity for each session. This may be achieved by the inclusion of session identities in information exchanged between the 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. The selection of parameters for both cases is discussed in Annex C.

NOTE 1 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 could allow an unauthorized 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 with $q$ 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 $F(q)$;

— $k$ the value $(q - 1)/r$, also called the cofactor, satisfying $k = 2$ or $k = 2p_1p_2...p_t$, for primes $p_i > r$, $i = 1, 2, ..., t$.

A method of generating $g_{q-1}$ can be found in Chapter 4 of References [18] and [22].

A set of EC domain parameters consists of:

— $F(q)$ a specific representation of the finite field with $q$ 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 $a_2$. For the purpose of this document, an elliptic curve is not only the set of points on the curve, but also a group operation defined on these points as specified in ISO/IEC 15946-1[13]. It is defined by one of the following two formulae:

  — $Y^2 = X^3 + a_1X + a_2$ 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 $k = 2^n$ or $k = 2^np_1p_2...p_t$, for $n = \{0, 1, 2\}$ and primes $p_i > r$, $i = 1, 2, ..., t$.

NOTE 2 If the cofactor $k$ is $2^np_1p_2...p_t$, for $n = \{0, 1, 2\}$ and primes $p_i < r$, $i = 1, 2, ..., t$, and the key token check function $T_{EC}$ in 6.2.3.3 is used, small subgroup attacks (e.g. see Reference [21]) are possible.

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. For further information on point representations, see ISO/IEC 15946-1[13].

In many of the mechanisms specified in this document, one of the participants is required to select a value at random from a given set of values, e.g. an integer from a specified range. This shall be implemented using a random number generation method based on a random bit generation (e.g. one of the methods in ISO/IEC 18031[14]) in such a way that the selected value is chosen uniformly (or near uniformly) at random from the complete set of possible values. Furthermore, when prime numbers are generated, it is assumed that the generation follows a secure method, e.g., using one of the methods in ISO/IEC 18032[15].

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[11].

# 6 Password-authenticated key agreement

## 6.1 General

Clause 6 specifies five password-authenticated key agreement mechanisms. The first and second mechanisms, specified in 6.2 and 6.3 respectively, are balanced password-authenticated key agreement mechanisms, which require the two entities to share a weak secret. The third, fourth, and fifth mechanisms, specified in 6.4, 6.5, and 6.6, 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 five 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:**

a) *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.

b) *Check validity of key tokens.* Depending on the operations for producing key tokens in step a), 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.

c) *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.

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 Balanced Mechanisms 1 and 2 but mandatory in Augmented Mechanisms 1, 2, and 3.

## 6.2 Balanced Key Agreement Mechanism 1 (BKAM1)

### 6.2.1 General

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, $\pi$. This mechanism provides mutual implicit key authentication and, optionally, mutual explicit key authentication.

This mechanism works in both the DL and EC settings.

NOTE This mechanism is based on References [6] and [17] and the mechanism called {DL,EC}BPKAS-SPEKE in Reference [8].



 

### 6.2.2 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, $\pi$;

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

— a random element derivation function, $R$;

— a key token generation function, $D$;

— a key token check function, $T$;

— a session identity generation function, $S$;

— 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 $B$ shall 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. A more detailed discussion of cofactor multiplication can be found in Reference [12].

### 6.2.3 Functions

#### 6.2.3.1 Random element derivation function, *R*

The random element derivation function, $R$, takes an octet string, $x$, as input and produces a selected group element written $R(x)$ as output. Balanced Key Agreement Mechanism 1 can be used with any 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 Formula (1):

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

— $R_{1EC}$ 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 Formula (2):

$$R_{1EC}(x) = \text{I2P}(\text{BS2I}(H(x))) \tag{2}$$

— $R_{2DL}$ 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 Formula (3):

$$R_{2DL}(x) = g_a * g_b^{\text{BS2I}(H(x))} \bmod q \tag{3}$$

— $R_{2EC}$ 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 Formula (4):

$$R_{2EC}(x) = G_a + [\text{BS2I}(H(x))] \times G_b. \quad (4)$$

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, even if operation of the mechanism is not terminated, there is no known 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, $T$.

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

NOTE 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 $R_2$.

#### 6.2.3.2 Key token generation function, *D*

The key token generation function, $D$, takes an integer $x$ and a group element $y$ as input and produces another group element written $D(x, y)$ as output. Balanced Key Agreement Mechanism 1 can be used with either 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 Formula (5):

$$D_{DL}(x, y) = y^x \bmod q \quad (5)$$

— $D_{EC}$ 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 Formula (6):

$$D_{EC}(x, Y) = [x] \times Y \quad (6)$$

#### 6.2.3.3 Key token check function, *T*

The key token check function, $T$, takes a group element $x$ as input and produces a Boolean value written $T(x)$ as output. Balanced Key Agreement Mechanism 1 can be used with either 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 $k = 2^n p_1 p_2 ... p_t$ 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, $T_{EC}(X) = 1$.

NOTE By checking $[2^n] \times X \neq 0_E$, one can avoid a data string $X$ whose order is smaller than $r$.

#### 6.2.3.4 Session identity generation function, *S*

The session identity generation function, $S$, takes as input two distinguishing identities of entities, represented as octet strings $A$ and $B$, two group elements $x$ and $y$, and optionally two texts represented as octet strings *text1* and *text2*, and produces an integer written $S(A, B, x, y, text1, text2)$ as output. The function $S$ is defined as Formulae (7), (8), and (9):

$$s_A = \text{BS2I}(H(A||\text{GE2OS}_x(x)||text1)) \tag{7}$$

$$s_B = \text{BS2I}(H(B||\text{GE2OS}_x(y)||text2)) \tag{8}$$

$$S(A, B, x, y, text1, text2) = MAX(s_A, s_B)||MIN(s_A, s_B) \tag{9}$$

#### 6.2.3.5 Secret value derivation function, *V*

The secret value derivation function, $V$, takes 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. Balanced 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 $\{1, ..., r - 1\}$, $y$ from $\{2, ..., q - 2\}$, and $b$ from $\{0, 1\}$, $V_{DL}$ is defined as Formula (10):

$$V_{DL}(x, y, b) = y^{x * k^b} \bmod q \tag{10}$$

— $V_{EC}$ 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 $\{1, ..., r - 1\}$, a point $Y$ ($\neq 0_E$) on the curve $E$ and $b$ from $\{0, 1\}$, $V_{EC}$ is defined as Formula (11):

$$V_{EC}(x, Y, b) = [k^b * x] \times Y \tag{11}$$

#### 6.2.3.6 Key derivation function, *K*

The key derivation function $K$ takes 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 $\{P_1, P_2, ...\}$ as input, and produces a bit string written $K(x, P, L_K)$ as output. Balanced 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 Formula (12):

$$K(x, P, L_K) = h(x||P, L_K) \tag{12}$$

NOTE 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.

### 6.2.4 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.

**Key token construction (A1)**

$A$ performs the following steps:

— compute $g_1 = R(\pi)$ as the base of its key token;

— choose an integer $s_A$ randomly from $\{1, ..., r - 1\}$ 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 the 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 method (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 $sID = S(A, B, w_A, w_B, null, null)$ as the session identity;

— compute $K_i = K(\text{I2OS}(sID)||\text{GE2OS}_X(z), P_i, L_K)$ for each key derivation parameter, $P_i$ as a shared secret key.

**Shared secret key derivation method (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 $sID = S(B, A, w_B, w_A, null, null)$ as the session identity;

— compute $K_i = K(\text{I2OS}(sID)||\text{GE2OS}_X(z), P_i, L_K)$ for each key derivation parameter, $P_i$ as a shared secret key.

NOTE 1 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 happen after A1 and B1.



 

NOTE 2 The session identity added into the key derivation function input is used to avoid the impersonation and the key-malleability attacks addressed in Reference [6].

**Key confirmation method (A3 and B3) (optional)**

*A* performs the following steps (A3):

— compute $o_A = H(A||B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z)||\text{GE2OS}_X(g_1))$;

— make $o_A$ available to *B*.

*B* performs the following steps (B3):

— receive $o_A$ from *A*;

— compute $o_A' = H(A||B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z)||\text{GE2OS}_X(g_1))$;

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

**Key confirmation method (B4 and A4) (optional)**

*B* performs the following steps (B4):

— compute $o_B = H(B||A||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(z)||\text{GE2OS}_X(g_1))$;

— make $o_B$ available to *A*.

*A* performs the following steps (A4):

— receive $o_B$ from *B*;

— compute $o_B' = H(B||A||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(z)||\text{GE2OS}_X(g_1))$;

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

Function $\text{GE2OS}_X$ (Group Element to Octet String conversion) is described in Annex A.

NOTE 3 Entities *A* and *B* are free to choose A3 and B3, or B4 and A4.

## 6.3 Balanced Key Agreement Mechanism 2 (BKAM2)

### 6.3.1 General

This 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, $\pi$. This mechanism provides mutual implicit key authentication and, optionally, mutual explicit key authentication. In this mechanism, it is assumed that the octet string, $\pi$, is derived from a password, e.g. a password in its plaintext form or a hash of the password. Additional data (e.g. an entity identity, a session identity and/or a salt) may be optionally included in the definition of the octet string, $\pi$.

This mechanism works in both the DL and EC settings.

NOTE This mechanism is based on the J-PAKE scheme of Reference [5]. An independent analysis of J-PAKE with security proofs in a formal model with algebraic adversaries and random oracles can be found in Reference [1].

### 6.3.2 Prior shared parameters

Key agreement between two entities *A* and *B* takes place in an environment in which the two entities share the following parameters:

— a shared password-based octet string, $\pi$;

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

— a random key token factor generation function, $R$;

— a random non-identity key token factor generation function, $N$;

— a non-identity key token check function, $U$;

— a key token generation function, $D$;

— a zero-knowledge proof generation function, $Z$;

— a zero-knowledge proof check function, $M$;

— a key token combination function, $C$;

— a secret value derivation function, $V$;

— 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, $L_K$.

### 6.3.3 Functions

#### 6.3.3.1 A random key token factor generation function, *R*

The random key token factor generation function, $R$, produces a random key token factor as output. In the DL setting, it returns a random value from $\{0, ..., r - 1\}$. In the EC setting, it returns a random value from $\{1, ..., r - 1\}$.

#### 6.3.3.2 A random non-identity key token factor generation function, *N*

The random non-identity key token factor generation function $N$ produces a random non-identity key token factor as output. In the DL setting, it returns a random value from $\{1, ..., r - 1\}$. In the EC setting, it returns a random value from $\{1, ..., r - 1\}$.

#### 6.3.3.3 A non-identity key token check function, *U*

The non-identity key token check function, $U$, takes a group element $y$ as input and produces a Boolean value written $U(y)$ as output. In the DL setting, $U_{DL}(y)$ returns 1 if $y \neq 1$ and 0 otherwise. In the EC setting, $U_{EC}(y)$ returns 1 if $y \neq 0_E$ and 0 otherwise.

#### 6.3.3.4 Key token generation function, *D*

The key token generation function, $D$, takes an integer $x$ and a group element $y$ as input and produces another group element written $D(x, y)$ as output. Balanced Key Agreement Mechanism 2 can be used with either 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 $\{0, ..., r - 1\}$ and an integer $y$, the generator of the group, $D_{DL}$, is defined as Formula (13):

$$D_{DL}(x, y) = y^x \bmod q \tag{13}$$



 

— $D_{EC}$ 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 generator of the group, $D_{EC}$ is defined as Formula (14):

$$D_{EC}(x, Y) = [x] \times Y \tag{14}$$

#### 6.3.3.5 A zero-knowledge proof generation function, *Z*

The zero-knowledge proof generation function, $Z$, takes an integer, two group elements and an entity identity as input and produces a tuple that consists of a group element and an integer as output. Balanced Key Agreement Mechanism 2 can be used with either of the following $Z$ functions, $Z_{DL}$ and $Z_{EC}$:

— $Z_{DL}$ 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 five inputs, namely, an integer $x$ from $\{0, ..., r - 1\}$, a group element $y$ as the generator of the group, a group element $u$ from the output of the function $D(x, y)$, the identity of the user who generates the zero-knowledge proof and an optional octet string *text*, $Z_{DL}$ chooses a random integer $v$ from $\{0, ..., r - 1\}$, computes $W = y^v \bmod q$ and returns Formula (15):

$$Z_{DL}(x, y, u, ID, text) = (W, t) \tag{15}$$

where $t = v - x * c \bmod r$ and $c = H(\text{GE2OS}_X(y)||\text{GE2OS}_X(W)||\text{GE2OS}_X(u)||ID||text)$.

— $Z_{EC}$ 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 five inputs, namely an integer $x$ from $\{1, ..., r - 1\}$, a point $Y$ as the generator of the group, a group element $U$ from the output of the function $D(x, Y)$, the identity of the user who generates the zero-knowledge proof and an optional octet string *text*, $D_{EC}$ chooses a random integer $v$ from $\{1, ..., r - 1\}$, computes $W = [v] \times Y$ and returns Formula (16):

$$Z_{EC}(x, Y, U, ID, text) = (W, t) \tag{16}$$

where $t = v - x * c \bmod r$ and $c = H(\text{GE2OS}_X(Y)||\text{GE2OS}_X(W)||\text{GE2OS}_X(U)||ID||text)$.

NOTE The optional *text* is included in the definition above for flexibility, as the same function can be useful in other security applications, e.g. to prove the possession of a long-term private key during the public key registration procedure where more contextual information such as the issuer name, timestamps and so on needs to be included in the hash function.

#### 6.3.3.6 A zero-knowledge proof check function, *M*

The zero-knowledge proof check function, $M$, takes a group element, a tuple from the output of the $Z$ function, a generator of the group, and an entity identity as input and produces a Boolean value as output. Balanced Key Agreement Mechanism 2 can be used with either of the following two functions, $M_{DL}$ and $M_{EC}$:

— $M_{DL}$ 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 six inputs, namely, a group element $x$, a tuple $(W, t)$ from the output of the $Z$ function, a generator $g$, the identity $ID$ of the entity who generated the zero-knowledge proof and an optional octet string *text*, $M_{DL}$ is defined as follows:

  — if $x$ does not represent an integer, $M_{DL}(x, W, t, g, ID, text) = 0$;

  — if $x \leq 0$, $M_{DL}(x, W, t, g, ID, text) = 0$;

  — if $x \geq q - 1$, $M_{DL}(x, W, t, g, ID, text) = 0$;

- if $x^r \neq 1 \bmod q$, $M_{DL}(x, W, t, g, ID, text) = 0$;
- if $g^t * x^c \neq W \bmod q$ where $c = H(\text{GE2OS}_X(g)||\text{GE2OS}_X(W)||\text{GE2OS}_X(x)||ID||text)$, $M_{DL}(x, W, t, g, ID, text) = 0$;
- else, $M_{DL}(x, W, t, g, ID, text) = 1$.

— $M_{EC}$ 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 $0_E$), the cofactor $k$, and six inputs, namely a group element $X$, a tuple $(W, t)$ from the output of the $Z$ function, a generator $G$, the identity $ID$ of the entity who generated the zero-knowledge proof and an optional octet string $text$, $M_{EC}$ is defined as follows:

- if $X$ does not represent a point on $E$, $M_{EC}(X, W, t, G, ID, text) = 0$;
- if $[k] \times X = 0_E$, $M_{EC}(X, W, t, G, ID, text) = 0$;
- if $[t] \times G + [c] \times X \neq W$ where $c = H(\text{GE2OS}_X(G)||\text{GE2OS}_X(W)||\text{GE2OS}_X(X)||ID||text)$, $M_{EC}(X, W, t, G, ID, text) = 0$;
- else, $M_{EC}(X, W, t, G, ID, text) = 1$.

NOTE The zero-knowledge proof check function defined in the DL setting does not exclude $x = 1$. In applications where the input $x$ is not allowed to be the group identity, there should be an additional check to ensure $x \neq 1$.

#### 6.3.3.7 A key token combination function, *C*

The key token combination function, $C$, takes three key tokens as input and produces a new key token as output. Balanced Key Agreement Mechanism 2 can be used with either of the following two 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 $q$), three inputs, namely, group elements $x$, $y$ and $z$, $C_{DL}$ is defined as Formula (17):

$$C_{DL}(x, y, z) = x * y * z \bmod q \tag{17}$$

— $C_{EC}$ 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 as input the EC domain parameters and three group elements $X$, $Y$ and $Z$, $C_{EC}$ is defined as Formula (18):

$$C_{EC}(X, Y, Z) = X + Y + Z \tag{18}$$

#### 6.3.3.8 Secret value derivation function, *V*

The secret value derivation function, $V$, takes a group element $x$, a group element $y$, an integer $c$ and an integer $d$ as input and produces another group element written $V(x, y, c, d)$ as output. Balanced Key Agreement Mechanism 2 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 four inputs, a group element $x$, a group element $y$, an integer $c$ from $\{1, ..., r - 1\}$ and an integer $d$ from $\{1, ..., r - 1\}$, $V_{DL}$ is defined as Formula (19):

$$V_{DL}(x, y, c, d) = (x / y^c)^d \bmod q \tag{19}$$

— $V_{EC}$ 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 four inputs, a point $X$ on the curve, a point $Y$ on the curve, an integer $c$ from {1, ..., $r$ − 1} and an integer $d$ from {1, ..., $r$ − 1}, $V_{EC}$ is defined as Formula (20):

$$V_{EC}(X, Y, c, d) = [d] \times (X - [c] \times Y) \quad (20)$$

#### 6.3.3.9 Key derivation function, *K*

The key derivation function, $K$, takes an octet string $x$, a length (in bits) $L_K$ of output of function $K$, and a key derivation parameter octet string $P$ from {$P_1$, $P_2$, ...} as input, and produces a bit string written $K(x, P, L_K)$ as output. Balanced Key Agreement Mechanism 2 makes use of a one-way function as Function $K$, i.e. given $x$, $P$ and $L_K$ as input, $K$ is defined as Formula (21):

$$K(x, P, L_K) = h(x \,||\, P, L_K) \quad (21)$$

NOTE 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 key for a symmetric cipher, the value of $L_K$ is the key length of a specific symmetric cipher mechanism.

### 6.3.4 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 A4 and B4 are optional.

**Round-1 key token construction (A1)**

$A$ performs the following steps:

— choose a random integer $x_{A1}$ as its first key token factor by calling the $R$ function;

— choose a random integer $x_{A2}$ its second key token factor by calling the $N$ function;

— compute $X_{A1} = D(x_{A1}, g)$ as the first key token with $g$ being the generator;

— compute $X_{A2} = D(x_{A2}, g)$ as the second key token with $g$ being the generator;

— compute $(W_{A1}, t_{A1}) = Z(x_{A1}, g, X_{A1}, A, null)$ as the zero-knowledge proof for the knowledge of $x_{A1}$ for the key token $X_{A1}$;

— compute $(W_{A2}, t_{A2}) = Z(x_{A2}, g, X_{A2}, A, null)$ as the zero-knowledge proof for the knowledge of $x_{A2}$ for the key token $X_{A2}$;

— make $X_{A1}$, $X_{A2}$, $(W_{A1}, t_{A1})$ and $(W_{A2}, t_{A2})$ available to $B$.

**Round-1 Key token construction (B1)**

$B$ performs the following steps:

— choose a random integer $x_{B1}$ as its first key token factor by calling the $R$ function;

— choose a random integer $x_{B2}$ as its second key token factor by calling the $N$ function;

— compute $X_{B1} = D(x_{B1}, g)$ as the first key token with $g$ being the generator;

— compute $X_{B2} = D(x_{B2}, g)$ as the second key token with $g$ being the generator;

— compute $(W_{B1}, t_{B1}) = Z(x_{B1}, g, X_{B1}, B, null)$ as the zero-knowledge proof for the knowledge of $x_{B1}$ for the key token $X_{B1}$;

— compute $(W_{B2}, t_{B2}) = Z(x_{B2}, g, X_{B2}, B, null)$ as the zero-knowledge proof for the knowledge of $x_{B2}$ for the key token $X_{B2}$;

— make $X_{B1}$, $X_{B2}$, $(W_{B1}, t_{B1})$ and $(W_{B2}, t_{B2})$ available to $B$.

**Round-2 key token construction (A2)**

*A* performs the following steps:

— receive $X_{B1}$, $X_{B2}$, ($W_{B1}$, $t_{B1}$), and ($W_{B2}$, $t_{B2}$) from *B*;

— check that $X_{B2}$ is not an identity element using $U(X_{B2})$: if $U(X_{B2})$ returns 0, output "invalid" and stop; otherwise, carry on;

— check validity of $X_{B1}$ and ($W_{B1}$, $t_{B1}$) using $M(X_{B1}, W_{B1}, t_{B1}, g, B, null)$: if $M(X_{B1}, W_{B1}, t_{B1}, g, B, null)$ returns 0, output "invalid" and stop; otherwise, carry on;

— check validity of $X_{B2}$ and ($W_{B2}$, $t_{B2}$) using $M(X_{B2}, W_{B2}, t_{B2}, g, B, null)$: if $M(X_{B2}, W_{B2}, t_{B2}, g, B, null)$ returns 0, output "invalid" and stop; otherwise, carry on;

— compute $g_A = C(X_{A1}, X_{B1}, X_{B2})$ as a new generator;

— check that $g_A$ is not an identity element using $U(g_A)$: if $U(g_A)$ returns 0, output "invalid" and stop; otherwise, carry on;

— compute $x_{A3}$ = BS2I($\pi$) * $x_{A2}$ mod $r$ as a new key token factor;

— compute $X_{A3} = D(x_{A3}, g_A)$ as a new key token with $g_A$ being the generator;

— compute $(W_{A3}, t_{A3}) = Z(x_{A3}, g_A, X_{A3}, A, null)$ as the zero-knowledge proof for the knowledge of $x_{A3}$ for the key token $X_{A3}$ with $g_A$ being the generator;

— make $X_{A3}$ and ($W_{A3}$, $t_{A3}$) available to *B*.

**Round-2 Key token construction (B2)**

*B* performs the following steps:

— receive $X_{A1}$, $X_{A2}$, ($W_{A1}$, $t_{A1}$) and ($W_{A2}$, $t_{A2}$) from *A*;

— check that $X_{A2}$ is not an identity element using $U(X_{A2})$: if $U(X_{A2})$ returns 0, output "invalid" and stop; otherwise, carry on;

— check validity of $X_{A1}$ and ($W_{A1}$, $t_{A1}$) using $M(X_{A1}, W_{A1}, t_{A1}, g, A, null)$: if $M(X_{A1}, W_{A1}, t_{A1}, g, A, null)$ returns 0, output "invalid" and stop; otherwise, carry on;

— check validity of $X_{A2}$ and ($W_{A2}$, $t_{A2}$) using $M(X_{A2}, W_{A2}, t_{A2}, g, A, null)$: if $M(X_{A2}, W_{A2}, t_{A2}, g, A, null)$ returns 0, output "invalid" and stop; otherwise, carry on;

— compute $g_B = C(X_{B1}, X_{A1}, X_{A2})$ as a new generator;

— check that $g_B$ is not an identity element using $U(g_B)$: if $U(g_B)$ returns 0, output "invalid" and stop; otherwise, carry on;

— compute $x_{B3}$ = BS2I($\pi$) * $x_{B2}$ mod $r$ as a new key token factor;

— compute $X_{B3} = D(x_{B3}, g_B)$ as a new key token with $g_B$ being the generator;

— compute $(W_{B3}, t_{B3}) = Z(x_{B3}, g_B, X_{B3}, B, null)$ as the zero-knowledge proof for the knowledge of $x_{B3}$ for the key token $X_{B3}$ with $g_B$ being the generator;

— make $X_{B3}$ and ($W_{B3}$, $t_{B3}$) available to *B*.

**Shared secret key derivation (A3)**

*A* performs the following steps:

— receive $X_{B3}$ and ($W_{B3}$, $t_{B3}$) from *B*;

— compute $g_B = C(X_{B1}, X_{A1}, X_{A2})$ as a new generator;

— check that $g_B$ is not an identity element using $U(g_B)$: if $U(g_B)$ returns 0, output "invalid" and stop; otherwise, carry on;

— check validity of $X_{B3}$ and $(W_{B3}, t_{B3})$ using $M(X_{B3}, W_{B3}, t_{B3}, g_B, B, null)$: if $M(X_{B3}, W_{B3}, t_{B3}, g_B, B, null) = 0$, output "invalid" and stop; otherwise, carry on;

— compute $z = V(X_{B3}, X_{B2}, x_{A3}, x_{A2})$ as a shared secret value,

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

**Shared secret key derivation (B3)**

*B* performs the following steps:

— receive $X_{A3}$ and $(W_{A3}, t_{A3})$ from *A*;

— compute $g_A = C(X_{A1}, X_{B1}, X_{B2})$ as a new generator;

— check that $g_A$ is not an identity element using $U(g_A)$: if $U(g_A)$ returns 0, output "invalid" and stop; otherwise, carry on;

— check validity of $X_{A3}$ and $(W_{A3}, t_{A3})$ using $M(X_{A3}, W_{A3}, t_{A3}, g_A, A, null)$: if $M(X_{A3}, W_{A3}, t_{A3}, g_A, A, null) = 0$, output "invalid" and stop; otherwise, carry on;

— compute $z = V(X_{A3}, X_{A2}, x_{B3}, x_{B2})$ as a shared secret value;

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

In the key confirmation method defined below, it is assumed that a set of secret keys has been derived (following the steps in A3 and B3) by varying the key derivation parameter, $P_i$. Among the keys derived by *A*, one key is for explicit key confirmation and is denoted as $K_{AC}$; similarly, the key derived by *B* for explicit key confirmation is denoted as $K_{BC}$. Both keys are used in the key confirmation method below.

**Key confirmation (A4 and B4) (optional)**

*A* performs the following steps (A4):

— compute $o_A = mac(K_{AC}, KC\_1\_U||A||B||\text{GE2OS}_X(X_{A1})||\text{GE2OS}_X(X_{A2})||\text{GE2OS}_X(X_{B1})||\text{GE2OS}_X(X_{B2}))$;

— make $o_A$ available to *B*.

*B* performs the following steps (B4):

— receive $o_A$ from *A*;

— compute $o_A' = mac(K_{BC}, KC\_1\_U||A||B||\text{GE2OS}_X(X_{A1})||\text{GE2OS}_X(X_{A2})||\text{GE2OS}_X(X_{B1})||\text{GE2OS}_X(X_{B2}))$;

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

**Key confirmation (B5 and A5) (optional)**

*B* performs the following steps (B5):

— compute $o_B = mac(K_{BC}, KC\_1\_U||B||A||\text{GE2OS}_X(X_{B1})||\text{GE2OS}_X(X_{B2})||\text{GE2OS}_X(X_{A1})||\text{GE2OS}_X(X_{A2}))$, and

— make $o_B$ available to *A*.

*A* performs the following steps (A5):

— receive $o_B$ from *B*;

— compute $o_B' = mac(K_{AC}, KC\_1\_U||B||A||\text{GE2OS}_X(X_{B1})||\text{GE2OS}_X(X_{B2})||\text{GE2OS}_X(X_{A1})||\text{GE2OS}_X(X_{A2}))$;

— 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 Entities $A$ and $B$ are free to choose A4 and B4, or B5 and A5.

## 6.4 Augmented Key Agreement Mechanism 1 (AKAM1)

### 6.4.1 General

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, $\pi$, and $B$ has password verification data $v$ corresponding to $\pi$. This mechanism provides unilateral explicit key authentication, and optionally mutual key authentication.

This mechanism works in the DL setting.

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

NOTE 2 This mechanism is based on Reference [27] and the mechanism called DLAPKAS-SRP6 in Reference [8].

### 6.4.2 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 $g_{q\text{-}1}$ and $q$, specified in Clause 5;

— a password-based octet string $\pi$ 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 $B$ shall agree to use the same $P_i$ value;

— an integer, $c$, defined as $c = (\text{BS2I}(H(\text{I2OS}(g_{q\text{-}1})||\text{I2OS}(q))) \bmod q$;

— 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.



 

### 6.4.3 Functions

#### 6.4.3.1 Password verification element derivation function, *J*

The password verification element derivation function, *J*, takes a password-based octet string, $\pi$, as input and produces an integer written $J(\pi)$ as output. Augmented Key Agreement Mechanism 1 can be used with the following *J* function:

— Given DL domain parameters (including $g_{q\text{-}1}$ and $q$) and a password-based octet string input $\pi$, *J* is defined as Formula (22):

$$J(\pi) = g_{q\text{-}1}^{\mathrm{BS2I}(H(\pi))} \bmod q \tag{22}$$

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

#### 6.4.3.2 Key token generation function *D*

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

— Given DL domain parameters (including $g_{q\text{-}1}$ and $q$) and an input $x$ from $\{1, \ldots, q-2\}$, *D* is defined as Formula (23):

$$D(x) = g_{q\text{-}1}^{x} \bmod q \tag{23}$$

#### 6.4.3.3 Password-entangled key token generation function *C*

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

Given the DL domain parameters (including $g_{q\text{-}1}$ and $q$), and three inputs, the integer $c$, $x$ from $\{1, \ldots, q-2\}$ and the output of *J* function $v$, *C* is defined as Formula (24):

$$C(x, v, c) = v * c + g_{q\text{-}1}^{x} \bmod q \tag{24}$$

#### 6.4.3.4 Secret value derivation functions, $V_A$ and $V_B$

a) The secret value derivation function, $V_A$, takes six inputs, the integer $c$, a password-based octet string $\pi$, an integer $x_A$ from $\{1, \ldots, q-2\}$, 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.

b) The secret value derivation function, $V_B$, takes four inputs, an integer $x_B$ from $\{1, \ldots, q-2\}$, 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.

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

Augmented Key Agreement Mechanism 1 can be used with the following $V_A$ and $V_B$ functions:

a) Given the DL domain parameters (including $g_{q\text{-}1}$ and $q$), the integer $c$, a password-based octet string $\pi$, an integer $x_A$ from {1, ..., $q$ – 2} , an output of $J$ function $v$, an output of $D$ function $y_A$, and an output of $C$ function $y_B$, $V_A$ is defined in the following steps:

   — compute $u_1$ = BS2I($H(\pi)$);

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

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

b) Given the DL domain parameters (including $g_{q\text{-}1}$ 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($y_B$)));

   — 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.

#### 6.4.3.5 Key derivation function, *K*

The key derivation function $K$ is the same as defined in 6.2.3.6.

### 6.4.4 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 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;

— 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;

— 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;

— compute $K_i = K(\text{I2OS}(z), P_i, L_K)$ 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;

— compute $K_i = K(\text{I2OS}(z), P_i, L_K)$ as a shared secret key for each key derivation parameter, $P_i$.

**Key confirmation (A3 and B3) (mandatory)**

*A* performs the following steps:

— compute $o_A = H(\text{I2OS}(4)||\text{I2OS}(w_A)||\text{I2OS}(w_B)||\text{I2OS}(z)||\text{I2OS}(v))$;

— make $o_A$ available to *B*.

*B* performs the following steps:

— receive $o_A$ from *A*;

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

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

**Key confirmation (B4 and A4) (optional)**

*B* performs the following steps:

— compute $o_B = H(\text{I2OS}(3)||\text{I2OS}(w_A)||\text{I2OS}(w_B)||\text{I2OS}(z)||\text{I2OS}(v))$;

— make $o_B$ available to *A*.

*A* performs the following steps:

— receive $o_B$ from *B*;

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

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

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

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

## 6.5 Augmented Key Agreement Mechanism 2 (AKAM2)

### 6.5.1 General

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 $\pi$ and *B* has password verification data *v* corresponding to $\pi$. This mechanism provides unilateral explicit key authentication, and optionally mutual key authentication.

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

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

NOTE 2 This mechanism is based on References [19] and [20] and the mechanism called {DL, EC}APKAS-AMP in Reference [8].

### 6.5.2 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, $\pi$, 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 $B$ shall agree to use the same $P_i$ value;

— the length of a shared secret key, $L_K$.

### 6.5.3 Functions

#### 6.5.3.1 Password verification element derivation function, *J*

The password verification element derivation function $J$ takes a password-based octet string, $\pi$, as input and produces a selected group element defined over $F(q)$ written $J(\pi)$ as output. Augmented Key Agreement Mechanism 2 can be used with either 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, $\pi$, $J_{DL}$ is defined as Formula (25):

$$J_{DL}(\pi) = g^{\mathrm{BS2I}(H(\pi))} \bmod q \tag{25}$$

— $J_{EC}$ 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, $\pi$, $J_{EC}$ is defined as Formula (26):

$$J_{EC}(\pi) = [\mathrm{BS2I}(H(\pi))] \times G \tag{26}$$

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



 

#### 6.5.3.2 Key token generation function, *D*

The key token generation function, *D*, takes an integer *x* from {1, ..., *r* − 1} as input, and produces a selected group element written *D*(*x*) as output. Augmented Key Agreement Mechanism 2 can be used with either 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 Formula (27):

$$D_{DL}(x) = g^x \bmod q \tag{27}$$

— $D_{EC}$ 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 Formula (28):

$$D_{EC}(x) = [x] \times G \tag{28}$$

#### 6.5.3.3 Password-entangled key token generation function, *C*

The password-entangled key token generation function, *C*, takes 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. Augmented Key Agreement Mechanism 2 can be used with either 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 $e = \text{BS2I}(H(\text{I2OS}(1)||\text{GE2OS}_X(y)))$;

  — 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 $e = \text{BS2I}(H(\text{I2OS}(1)||\text{GE2OS}_X(Y)))$;

  — 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 $\text{GE2OS}_X$ (Group Element to Octet String conversion) are described in Annex A.

#### 6.5.3.4 Key token check function, *T*

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

#### 6.5.3.5 Secret value derivation functions, $V_A$ and $V_B$

a) The secret value derivation function $V_A$ takes four inputs, a password-based octet string $\pi$, 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.

b) The secret value derivation function $V_B$ takes 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.

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

Augmented Key Agreement Mechanism 2 can be used with either of the following two $V_A$ functions, $V_{ADL}$ and $V_{AEC}$, and either of the following two $V_B$ functions, $V_{BDL}$ and $V_{BEC}$:

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 $q$), a password-based octet string $\pi$, 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($H(\pi)$)) mod $r$;

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

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

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 $F(q)$. 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 $V_{BDL}(x_B, y_A, y_B)$.

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 $r$), a password-based octet string $\pi$, an integer $x_A$ from {1, ..., $r$ − 1}, a point $Y_A$ (≠ $0_E$) on $E$, and a point $Y_B$ (≠ $0_E$) 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($H(\pi)$)) mod $r$;

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

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

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 $G$), an integer $x_B$ from {1, ..., $r$ − 1}, a point $Y_A$ (≠ $0_E$) on $E$, and a point $Y_B$ (≠ $0_E$) 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.

#### 6.5.3.6 Key derivation function, *K*

The key derivation function *K* is the same as defined in 6.2.3.6.

### 6.5.4 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 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;

— 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);

— 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 $z = V_A(T, s_A, w_A, w_B)$ as an agreed secret value;

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

**Shared secret key derivation (B2)**

*B* performs the following steps:

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

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

**Key confirmation (A3 and B3) (mandatory)**

*A* performs the following steps (A3):

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

— make $o_A$ available to *B*.

*B* performs the following steps (B3):

— receive $o_A$ from *A*;

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

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

**Key confirmation (B4 and A4) (optional)**

*B* performs the following steps (B4):

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

— make $o_B$ available to *A*.

*A* performs the following steps (A4):

— receive $o_B$ from *B*;

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

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

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

Function $\text{GE2OS}_X$ (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 Using the Pohlig-Hellman decomposition attack, the lowest one or two bits of *B*'s secret value $s_B$ could be discernable by an attacker, if *k* is divisible by 2 or 4.

## 6.6 Augmented Key Agreement Mechanism 3 (AKAM3)

### 6.6.1 General

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

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

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

NOTE 2 This mechanism is based on References [23] and [24].

### 6.6.2 Prior shared parameters

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) as specified in Clause 5;

— a password-based octet string $\pi$ 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 $B$ shall agree to use the same $P_i$ value;

— the length of a shared secret key, $L_K$.

### 6.6.3 Functions

#### 6.6.3.1 Password verification element derivation function, *J*

The password verification element derivation function, $J$, takes a password-based octet string, $\pi$, as input and produces a selected group element defined over $F(q)$ written $J(\pi)$ as output. Augmented Key Agreement Mechanism 3 can be used with either 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, $\pi$, $J_{DL}$ is defined as Formula (29):

$$J_{DL}(\pi) = g^{BS2I(H(\pi))} \bmod q \tag{29}$$

— $J_{EC}$ 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 $\pi$, $J_{EC}$ is defined as Formula (30):

$$J_{EC}(\pi) = [BS2I(H(\pi))] \times G \tag{30}$$

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

#### 6.6.3.2 Key token generation function, *D*

The key token generation function, $D$, takes an integer $x$ from $\{1, ..., r - 1\}$ as input, and produces a selected group element written $D(x)$ as output. Augmented Key Agreement Mechanism 3 can be used with either 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 Formula (31):

$$D_{DL}(x) = g^x \bmod q \tag{31}$$

— $D_{EC}$ 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 Formula (32):

$$D_{EC}(x) = [x] \times G \tag{32}$$

#### 6.6.3.3 Password-entangled key token generation function, *C*

The password-entangled key token generation function, $C$, takes 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. Augmented Key Agreement Mechanism 3 can be used with either 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)||$A$||$B$||GE2OS$_X$($y$)));

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

  — 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)||$A$||$B$||GE2OS$_X$($Y$)));

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

  — check if $[2^n] \times C_{EC}(x, V, Y) = 0_E$, 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 specified in Annex A.

#### 6.6.3.4 Key token check function, *T*

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

#### 6.6.3.5 Secret value derivation functions, $V_A$ and $V_B$

a) The secret value derivation function, $V_A$, takes four inputs, a password-based octet string $\pi$, 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.

b) The secret value derivation function $V_B$ takes one input, an integer $x_B$ from {1, ..., $r$ − 1}, and produces a selected group element written $V_B(x_B)$ as output.

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

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

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 $q$), a password-based octet string $\pi$, 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)||$A$||$B$||GE2OS$_X$($y_A$)));

  — compute $u = 1/(x_A + (\text{BS2I}(H(\pi)) * e)) \bmod r$;

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

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

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

— compute $V_{BDL}(x_B) = g^{x_B} \bmod q$;

— output $V_{BDL}(x_B)$.

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 $r$), a password-based octet string $\pi$, 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)||$A$||$B$||GE2OS$_X$($Y_A$)));

— compute $u = 1/(x_A + (\text{BS2I}(H(\pi)) * e)) \bmod r$;

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

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

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 $G$), and an integer $x_B$ from {1, ..., $r$ – 1}, $V_{BEC}$ is defined in the following steps:

— compute $V_{BEC}(x_B) = [x_B] \times G$;

— output $V_{BEC}(x_B)$.

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.

#### 6.6.3.6 Key derivation function, *K*

The key derivation function $K$ is the same as defined in 6.2.3.6.

### 6.6.4 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 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;

— 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);

— 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;

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

**Shared secret key derivation (B2)**

$B$ performs the following steps:

— compute $z = V_B(s_B)$ as an agreed secret value;

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

**Key confirmation (A3 and B3) (mandatory)**

$A$ performs the following steps (A3):

— compute $o_A = H(\text{I2OS}(2)||A||B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z))$;

— make $o_A$ available to $B$.

$B$ performs the following steps (B3):

— receive $o_A$ from $A$;

— compute $o_A' = H(\text{I2OS}(2)||A||B||\text{GE2OS}_X(w_A)||\text{GE2OS}_X(w_B)||\text{GE2OS}_X(z))$;

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

**Key confirmation (B4 and A4) (optional)**

$B$ performs the following steps (B4):

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

— make $o_B$ available to $A$.

$A$ performs the following steps (A4):

— receive $o_B$ from $B$;

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

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

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

Function $\text{GE2OS}_X$ (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 Using the Pohlig-Hellman decomposition attack, the lowest one or two bits of $B$'s secret value $s_B$ could be discernable by an attacker, if $k$ is divisible by 2 or 4.

NOTE 3 In this mechanism, $w_A$ and $V_B(s_B)$ can be computed before the key agreement operation.

# 7 Password-authenticated key retrieval

## 7.1 General

Clause 7 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 the role of a client and $B$ may play the role of a server.

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

**Initialization 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:**

a) *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 $A$.

b) *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.

c) *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.

## 7.2 Key Retrieval Mechanism 1 (KRM1)

### 7.2.1 General

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 Reference [4] and the mechanism called {DL,EC}PKRS-1 in Reference [8].

### 7.2.2 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, $\pi$, 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, $L_K$.

### 7.2.3 Functions

#### 7.2.3.1 Random element derivation function, *R*

This function is $R_{1DL}$ or $R_{1EC}$ as defined in 6.2.3.1.

#### 7.2.3.2 Key token generation function, *D*

The key token generation function, $D$, is the same as specified in 6.2.3.2.

#### 7.2.3.3 Key token check function, *T*

This function is the same as defined in 6.2.3.3.

#### 7.2.3.4 Secret value derivation function, *V*

The secret value derivation function, $V$, takes 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 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 Formula (33):

$$V_{DL}(x, y) = y^{x^{-1} \bmod r} \bmod q \tag{33}$$

— $V_{EC}$ 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 Formula (34):

$$V_{EC}(x, Y) = [x^{-1} \bmod r] \times Y \tag{34}$$



 

#### 7.2.3.5 Key derivation function, *K*

This function is the same as defined in 6.2.3.6.

### 7.2.4 Key retrieval operation

This mechanism involves *A* performing a sequence of up to two steps, numbered A1 and A2, and *B* performing one step, numbered B1.

**Key token construction (A1)**

*A* performs the following steps:

— compute $g_1 = R(\pi)$ as the base element of its key token;

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

— compute $w_A = D(s_A, g_1)$ as its key token;

— 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;

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

— 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 $K_i = K(\text{GE2OS}_X(z), P_i, L_K)$ for each key derivation parameter octet string $P_i$ in $\{P_1, P_2, ...\}$ as a secret key.

Function $\text{GE2OS}_X$ (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 $s_B$ could be discernable by an attacker, when *k* is divisible by 2 or 4.

# Annex A
(normative)

# Functions for data type conversion

## A.1 General

This annex specifies data type conversion functions which are used as part of key establishment mechanisms in this document.

## A.2 I2OS and OS2I

This clause 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.

a) Write $x$ in its unique $l$-digit base 256 representation, as shown in Formula (A.1):

$$x = x_{l-1}\, 256^{l-1} + x_{l-2}\, 256^{l-2} + \ldots + x_1\, 256 + x_0 \quad \text{(A.1)}$$

where $0 \le x_i < 256$.

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

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

For example, I2OS(10 945) = 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.

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

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

c) Output $y$.

For example, OS2I(2A C1) = 10 945.

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

## A.3 BS2I

This clause 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.

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

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