INTERNATIONAL STANDARD

ISO/IEC 15946-3

First edition
2002-12-01

Information technology — Security techniques — Cryptographic techniques based on elliptic curves —

Part 3: Key establishment

Technologies de l'information — Techniques de sécurité — Techniques cryptographiques basées sur les courbes elliptiques —

Partie 3: Établissement de clé

Reference number
ISO/IEC 15946-3:2002(E)

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

Published in Switzerland

Contents

1 Scope 1
2 Normative references 1
3 Terms and definitions 2
4 Symbols and abbreviated terms 4
5 Key derivation functions 5
6 Cofactor multiplication 5
7 Key commitment 6
8 Key agreement mechanisms 6
8.1 Common information 6
8.2 Non-interactive key agreement of Diffie-Hellman type (KANIDH) 7
8.2.1 Setup 7
8.2.2 Mechanism 7
8.2.3 Properties 7
8.3 Key agreement of ElGamal type (KAEG) 7
8.3.1 Setup 7
8.3.2 Mechanism 8
8.3.3 Properties 8
8.4 Key agreement of Diffie-Hellman type 8
8.4.1 Setup 8
8.4.2 Mechanism 8
8.4.3 Properties 9
8.5 Key agreement of Diffie-Hellman type with 2 key pairs (KADH2KP) 9
8.5.1 Setup 9
8.5.2 Mechanism 9
8.5.3 Properties 10
8.6 Key agreement of Diffie-Hellman type with 2 signatures and key confirmation (KADH2SKC) 10
8.6.1 Setup 10

8.6.2 Mechanism 10
8.6.3 Properties 11
9 Key agreement mechanisms not included in ISO/IEC 11770-3. 12
9.1 Common information 12
9.2 The Full Unified Model 12
9.2.1 Setup 12
9.2.2 Mechanism 12
9.2.3 Properties 13
9.3 Key agreement of MQV type with 1 pass (KAMQV1P) 13
9.3.1 Setup 13
9.3.2 Mechanism 13
9.3.3 Properties 14
9.4 Key agreement of MQV type with 2 passes (KAMQV2P) 14
9.4.1 Setup 14
9.4.2 Mechanism 14
9.4.3 Properties 15
10 Key transport mechanisms 15
10.1 Common information 15
10.2 Key transport of ElGamal type (KTEG) 15
10.2.1 Setup 15
10.2.2 Mechanism 16
10.2.3 Properties 16
10.3 Key transport of ElGamal type with originator signature (KTEGOS) 16
10.3.1 Setup 16
10.3.2 Mechanism 17
10.3.3 Properties 17
11 Key Confirmation 18
Annex A (informative) Examples of key derivation functions 20
A.1 The IEEE P1363 key derivation function 20
A.1.1 Preconditions 20
A.1.2 Input 20

A.1.3 Actions **20**

A.1.4 Output **20**

A.2 The ANSI X9.42 key derivation function **20**

A.2.1 Prerequisites **20**

A.2.2 Input **21**

A.2.3 Actions **21**

A.2.4 Output **22**

A.2.5 ASN.1 syntax **22**

A.3 The ANSI X9.63 key derivation function **22**

A.3.1 Prerequisites **23**

A.3.2 Input **23**

A.3.3 Actions **23**

A.3.4 Output **23**

Annex B (informative) **A comparison of the claimed properties of the mechanisms in this standard** **24**

B.1 Security Properties **24**

B.2 Performance Considerations **27**

Bibliography **29**

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 15946-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

ISO/IEC 15946 consists of the following parts, under the general title *Information technology — Security techniques — Cryptographic techniques based on elliptic curves*:

— *Part 1: General*

— *Part 2: Digital signatures*

— *Part 3: Key establishment*

— *Part 4: Digital signatures giving message recovery*

Annexes A and B of this part of ISO/IEC 15946 are for information only.

Introduction

Some of the most interesting and potentially useful public key cryptosystems that are currently available are cryptosystems based on elliptic curves defined over finite fields. The concept of an elliptic curve based public key cryptosystem is rather simple:

— Every elliptic curve is endowed with a binary operation "+" under which it forms a finite abelian group.

— The group law on elliptic curves extends in a natural way to a "discrete exponentiation" on the point group of the elliptic curve.

— Based on the discrete exponentiation on an elliptic curve one can easily derive elliptic curve analogues of the well known public key schemes of Diffie-Hellman and ElGamal type.

The security of such a public key system depends on the difficulty of determining discrete logarithms in the group of points of an elliptic curve. This problem is - with current knowledge - much harder than the factorization of integers or the computation of discrete logarithms in a finite field. Indeed, since Miller and Koblitz in 1985 independently suggested the use of elliptic curves for public key cryptographic systems, no substantial progress in tackling the elliptic curve discrete logarithm problem has been reported. In general, only algorithms that take exponential time are known to determine elliptic curve discrete logarithms. Thus, it is possible for elliptic curve based public key systems to use much shorter parameters than the RSA system or the classical discrete logarithm based systems that make use of the multiplicative group of some finite field. This yields significantly shorter digital signatures and system parameters and allows for computations using smaller integers.

This part of ISO/IEC 15946 describes schemes that can be used for key agreement and schemes that can be used for key transport. Where possible, the schemes are analogous to methods included in ISO/IEC 11770-3. Schemes that are not included in ISO/IEC 11770-3 are noted as such.

The International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) draw attention to the fact that it is claimed that compliance with this International Standard 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:

ISO/IEC JTC 1/SC 27 Standing Document 8 (SD 8)

SD 8 is publicly available at:
http://www.din.de/ni/sc27

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 15946 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.

Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 3: Key establishment

1 Scope

International Standard ISO/IEC 15946 specifies public key cryptographic techniques based on elliptic curves. The standard is split into four parts and includes the establishment of keys for secret key systems and digital signature mechanisms.

This part of ISO/IEC 15946 specifies techniques for key establishment, which includes key agreement and key transport, that use elliptic curves.

The scope of this standard is restricted to cryptographic techniques based on elliptic curves defined over finite fields of prime power order (including the special cases of prime order or characteristic two). The representation of elements of the underlying finite field is outside the scope of this standard. This standard does not fully specify the implementation of the techniques it defines. There may be different products that comply with this International Standard and yet are not compatible.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 15946. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 15946 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 9796 (parts 2 and 3), *Information technology — Security techniques — Digital signature schemes giving message recovery*

ISO/IEC 9797 (all parts), *Information technology — Security techniques — Message Authentication Codes (MACs)*

ISO/IEC 10118 (all parts), *Information technology — Security techniques — Hash-functions*

ISO/IEC 11770-3:1999, *Information technology — Security techniques — Key management — Part 3: Mechanisms using asymmetric techniques*

ISO/IEC 14888 (all parts), *Information technology — Security techniques — Digital signatures with appendix*

ISO/IEC 15946-1:2001, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 1: General*

ISO/IEC 15946-2:2001, *Information technology — Security techniques — Cryptographic techniques based on elliptic curves — Part 2: Digital signatures*

3 Terms and definitions

For the purposes of this part of ISO/IEC 15946, the definitions of Part 1 apply. In addition, the following definitions from ISO/IEC 11770-3 apply.

3.1 Asymmetric cryptographic technique: a cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key). The two transformations have the property that, given the public transformation, it is computationally infeasible to derive the private transformation.

3.2 Asymmetric encipherment system: a system based on asymmetric cryptographic techniques whose public transformation is used for encipherment and whose private transformation is used for decipherment.

3.3 Asymmetric key pair: a pair of related keys where the private key defines the private transformation and the public key defines the public transformation.

3.4 Signature system: a system based on asymmetric cryptographic techniques whose private transformation is used for signing and whose public transformation is used for verification.

3.5 Cryptographic check function: a cryptographic transformation which takes as input a secret key and an arbitrary string, and which gives a cryptographic check value as output. The computation of a correct check value without knowledge of the secret key shall be infeasible [ISO/IEC 9798-1:1997].

3.6 Cryptographic check value: information which is derived by performing a cryptographic transformation on the data unit [ISO/IEC 9798-4:1995].

NOTE The cryptographic check value is the output of the cryptographic check function.

3.7 Decipherment: the reversal of a corresponding encipherment [ISO/IEC 11770-1:1996].

3.8 Digital signature: data appended to, or a cryptographic transformation of, a data unit that allows the recipient of the data unit to prove the origin and integrity of the data unit and protect against forgery, e.g. by the recipient [ISO/IEC 11770-1:1996].

3.9 Distinguishing identifier: information which unambiguously distinguishes an entity [ISO/IEC 11770-1:1996].

3.10 Encipherment: the (reversible) transformation of data by a cryptographic algorithm to produce ciphertext, i.e. to hide the information content of the data [ISO/IEC 9798-1:1997].

3.11 Entity authentication: the corroboration that an entity is the one claimed [ISO/IEC 9798-1:1997].

3.12 Entity authentication of A to B: the assurance of the identity of entity A for entity B.

3.13 Explicit key authentication from A to B: the assurance for entity B that A is the only other entity that is in possession of the correct key.

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

3.14 Implicit key authentication from A to B: the assurance for entity B that A is the only other entity that can possibly be in possession of the correct key.

3.15 Key: a sequence of symbols that controls the operation of a cryptographic transformation (e.g. encipherment, decipherment, cryptographic check function computation, signature generation, signature verification, or key agreement) [ISO/IEC 11770-1:1996].

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

3.17 Key confirmation from A to B: the assurance for entity B that entity A is in possession of the correct key.

3.18 Key control: the ability to choose the key or the parameters used in the key computation.

3.19 Key establishment: the process of making available a shared secret to one or more entities. Key establishment includes key agreement and key transport.

3.20 Key token: key management message sent from one entity to another entity during the execution of a key management mechanism.

3.21 Key transport: the process of transferring a key from one entity to another entity, suitably protected.

3.22 Mutual entity authentication: entity authentication which provides both entities with assurance of each other's identity.

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

3.24 Private key: that key of an entity's asymmetric key pair which should only be used by that entity.

NOTE In the case of an asymmetric signature system the private key defines the signature transformation. In the case of an asymmetric encipherment system the private key defines the decipherment transformation.

3.25 Public key: that key of an entity's asymmetric key pair which can be made public

NOTE In the case of an asymmetric signature system the public key defines the verification transformation. In the case of an asymmetric encipherment system the public key defines the encipherment transformation. A key that is 'publicly known' is not necessarily globally available. The key may only be available to all members of a pre-specified group.

3.26 Secret key: a key used with symmetric cryptographic techniques by a set of specified entities.

3.27 Sequence number: a time variant parameter whose value is taken from a specified sequence which is non-repeating within a certain time period [ISO/IEC 11770-1:1996].

3.28 Time stamp: a data item which denotes a point in time with respect to a common reference [ISO/IEC 11770-1:1996].

The following definition from ISO/IEC 10118-1 applies.

3.29 hash-function: a function which maps strings of bits to fixed-length strings of bits, satisfying two properties.

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

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

NOTES

1 – The literature on this subject contains a variety of terms which have the same or similar meaning as hash-function. Compressed encoding and condensing function are some examples.

2 – Computational feasibility depends on the user's specific security requirements and environment.

Additional definitions which are required are as follows.

3.30 Forward secrecy with respect to *A*: the property that knowledge of A's long-term private key subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys.

3.31 Forward secrecy with respect to both A and B individually: the property that knowledge of A's long-term private key or knowledge of B's long-term private key subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys.

NOTE This differs from mutual forward secrecy in which knowledge of both A's and B's long-term private keys does not enable recomputation of previously derived keys.

3.32 Key derivation function: a key derivation function outputs one or more shared secrets, used as keys, given shared secrets and other mutually known parameters as input.

3.33 Mutual forward secrecy: the property that knowledge of both A's and B's long-term private keys subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys.

3.34 Prefix free representation: a representation of a data element for which concatenation with any other data does not produce a valid representation.

4 Symbols and abbreviated terms

Throughout this part of ISO/IEC 15946, the following symbols and notations are used in addition to those given in ISO/IEC 15946-1.

E_K	Symmetric encryption function using secret key *K*.
f	A cryptographic check function.
$f_K(Z)$	A cryptographic check value which is the result of applying the cryptographic check function f using as input a secret key *K* and an arbitrary data string *Z*.
h	The cofactor used in performing cofactor multiplication.
kdf	A key derivation function.
l	A supplementary value used in performing cofactor multiplication.
MAC	A Message Authentication Code algorithm.
MAC(K,Z)	A Message Authentication Code value which is the result of applying the MAC algorithm using as input the secret key *K* and an arbitrary data string *Z*.
parameters	Parameters used in the key derivation function.
S_X	Entity *X*'s private signature transformation.

NOTE No assumption is made on the nature of the signature transformation. In the case of a signature system with message recovery, $S_A(m)$ denotes the signature itself. In the case of a signature system with appendix, $S_A(m)$ denotes the message *m* together with the signature.

[|| *Text1*], [|| *Text2*]	Optional text, data or other information that may be included in a data block, if desired.
V_X	Entity *X*'s public signature verification transformation.
$\pi^*(\mathrm{Q})$	$(\pi(\mathrm{Q}) \bmod 2^{\lceil \rho/2 \rceil}) + 2^{\lceil \rho/2 \rceil}$ where $\rho = \lceil \log_2 n \rceil$.

5 Key derivation functions

The use of a shared secret as derived in Clauses 8 and 9 as a key for a symmetric cryptosystem without further processing is not recommended. It most often will be the case that the form of a shared secret will not conform to the form needed for a shared symmetric key, so some processing will be needed. The shared secret (often) has arithmetic properties and relationships that might result in a shared symmetric key not being chosen from the full key space. It is advisable to pass the shared secret through a key derivation function, which includes the use of a hash function. The use of an inadequate key derivation function compromises the security of the key agreement scheme in which it is used.

A key derivation function should produce keys that are computationally indistinguishable from randomly generated keys. The key derivation function should take as input the shared secret and a set of key derivation parameters and produce an output of the desired length.

Key derivation functions are also used in Clause 10 to encrypt the secret key before transport.

In order for the two parties in a key establishment mechanism to agree on a common secret key, the key derivation function must be agreed upon. The method of coming to such an agreement is outside the scope of this standard.

See Annex A for three examples of key derivation functions.

6 Cofactor multiplication

The key agreement mechanisms in Clauses 8 and 9 and the key transport mechanisms in Clause 10 require that the user's private key or key token be combined with another entity's public key or key token. If the other entity's public key or key token is not valid (i.e. it is not a point on the elliptic curve, or is not in the subgroup of order n) then performing this operation may result in some bits of the private key being leaked to an attacker. An example of this attack is the 'small subgroup attack'.

In order to prevent the 'small subgroup attack' and similar attacks, one option is to validate public keys and key tokens received from the other party using public key validation. See Part 1 of this standard for a description of public key validation.

As an alternative to verifying the order of the public key or key token, a technique called cofactor multiplication can be used. The values *h* and *l*, defined below, are used for cofactor multiplication in Clauses 8, 9 and 10.

If cofactor multiplication is desired, there are two options:

- If cofactor multiplication is used, and incompatibility with those not using cofactor multiplication is desired then let $h = \#E/n$ and $l = 1$. If this option is chosen, both parties involved must agree to use this option, otherwise the mechanism will not work.
- If cofactor multiplication is used, and compatibility with those not using cofactor multiplication is desired then let $h = \#E/n$ and $l = h^{-1} \bmod n$. When this option is chosen, the mechanisms described in Clauses 8 and 10 are consistent with ISO/IEC 11770-3.

  NOTE The value h^{-1} mod *n* will always exist since *n* is required to be greater than $4\sqrt{q}$ (see Part 1) and therefore gcd(*n*,*h*)=1.

If cofactor multiplication is not desired, there is one option:

- If cofactor multiplication is not used, then let $h = 1$ and $l = 1$. When this option is chosen, the mechanisms described in Clauses 8 and 10 are consistent with ISO/IEC 11770-3.

Regardless of whether or not cofactor multiplication is used and the type of compatibility that is chosen, if the shared key (or a component of the shared key) evaluates to the point at infinity ($\boldsymbol{0}_E$) then the user shall assume that the key agreement has failed.

It should be noted that it is most appropriate to perform these operations (public key validation or cofactor multiplication) if the other entity's public key or key token is not authenticated or the user's public key is long term. Performing public key validation for long-term keys and cofactor multiplication for ephemeral (short term) keys may also have performance advantages.

It should also be noted that if the other entity's public key is authenticated and the cofactor is small, then the amount of information that can be leaked is limited. Thus, it may not always be desired that these tests be performed.

7 Key commitment

Clauses 8 and 9 describe key agreement mechanisms where the established key is the result of applying a one-way function of the private key-agreement keys. However, one entity may know the other entity's public key or key token prior to choosing their private key. The entity can control the value of s bits in the established key, at the cost of generating 2^s candidate values for their private key-agreement key in the time interval between discovering the other entity's public key or key token and choosing their own private key [4].

A way to address this concern (if it is a concern) at the cost of one additional message/pass in the protocol is through the use of key commitment. Commitment can be done by having the first party hash the public key or key token and send the hash-code to the second party, the second party replies with his public key or key token and the first party replies with his public key or key token, which the second party can hash and verify that the result is equal to the hash-code sent earlier.

8 Key agreement mechanisms

Key agreement is the process of establishing a shared secret between two entities *A* and *B* in such a way that neither of them can predetermine the value of the shared secret.

Where authentic copies of the other party's public keys are required, they can be obtained using the techniques specified in ISO/IEC 11770-3, Clause 8.

NOTE All computations in this Clause must be performed in the order given by the formulae.

8.1 Common information

For all key agreement mechanisms, prior to the process of agreeing upon a shared secret, the following common information must be established between the parties and optionally validated (see Part 1 of this standard for a description of parameter validation):

— the elliptic curve parameters with which the key pairs shall be associated, which shall be the same for both parties' key pairs. This includes p, p^m or 2^m, a description of $\boldsymbol{F}(p)$, $\boldsymbol{F}(p^m)$ or $\boldsymbol{F}(2^m)$ and an indication of the basis used, $\boldsymbol{E}$, n and G.

In each of the mechanisms defined below, the resulting agreed key should not be used as a cryptographic key directly. Instead, it should be used as the input to a key derivation function, allowing both parties to derive the same cryptographic keys from it. Hence, it is also necessary for the two parties to agree on the following information:

— a key derivation function, *kdf*;
— any parameters to the key derivation function, and
— the type of cofactor multiplication that is to be performed (if any).

8.2 Non-interactive key agreement of Diffie-Hellman type (KANIDH)

This key agreement mechanism non-interactively establishes a shared secret between two entities *A* and *B*.

8.2.1 Setup

Prior to the process of agreeing upon a shared secret, in addition to the common information, the following must be established:

— for each entity *X*, a private key-agreement key d_X and a public key-agreement key P_X, which is an elliptic curve point satisfying $P_X=d_XG$. See Part 1 of this standard for a description of how to generate this key pair.

— for each entity, access to an authentic copy of the public key-agreement key of the other party.

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

8.2.2 Mechanism

8.2.2.1 Key construction (A): Entity *A* computes, using its own private key-agreement key d_A and *B*'s public key-agreement key P_B, the shared key as

$$K_{AB} = (d_A \cdot I)(hP_B).$$

8.2.2.2 Key construction (B): Entity *B* computes, using its own private key-agreement key d_B and *A*'s public key-agreement key P_A, the shared key as

$$K_{AB} = (d_B \cdot I)(hP_A).$$

8.2.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 0.

b) The mechanism provides mutual implicit key authentication.

NOTE As a consequence of the first property, the established secret between the same two users always has the same value. For this reason it is suggested that the input to the key derivation function in this case include time-varying information.

8.3 Key agreement of ElGamal type (KAEG)

This key agreement mechanism establishes a shared secret between two entities *A* and *B* in one pass.

8.3.1 Setup

Prior to the process of agreeing upon a shared secret, in addition to the common information, the following must be established:

— for entity *B*, a private key-agreement key d_B and a public key-agreement key P_B, which is an elliptic curve point satisfying $P_B=d_BG$. See Part 1 of this standard for a description of how to generate this key pair.

— for entity *A*, access to an authentic copy of the public key-agreement key of entity *B*.

Entity *A* should verify that entity *B*'s public key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

8.3.2 Mechanism

8.3.2.1 Key token construction (A): Entity *A* randomly and secretly generates *r* in the range {1,...,n-1}, computes *rG*, constructs the key token,

$$KT_{A1} = rG$$

and sends it to *B*.

8.3.2.2 Key construction (A): Entity *A* computes the shared key

$$K_{AB} = (r \cdot l)(hP_B).$$

8.3.2.3 Key construction (B): Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Using its own private key, entity *B* computes the shared key from KT_{A1} as follows:

$$K_{AB} = (d_B \cdot l)(hKT_{A1}).$$

8.3.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 1.

b) The mechanism provides implicit key authentication from *B* to *A* since *B* is the only entity other than *A* that can compute the shared secret.

c) The mechanism provides forward secrecy with respect to A.

8.4 Key agreement of Diffie-Hellman type

This key agreement mechanism establishes a shared secret between entities *A* and *B* in two passes.

8.4.1 Setup

This key agreement mechanism does not require any initial information other than the common information to be set up.

8.4.2 Mechanism

8.4.2.1 Key Token Construction

(A): Entity *A* randomly and secretly generates r_A in the range {1,...,n-1}, computes r_AG, constructs the key token,

$$KT_{A1} = r_AG$$

and sends it to *B*.

8.4.2.2 Key Token Construction (B): Entity *B* randomly and secretly generates r_B in the range {1,...,n-1}, computes r_BG, constructs the key token**,**

$$KT_{B1} = r_BG$$

and sends it to *A*.

8.4.2.3 Key Construction (A): Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *A* computes the shared key

$$K_{AB} = (r_A \cdot l)(hKT_{B1}).$$

8.4.2.4 Key Construction (B): Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *B* computes the shared key

$$K_{AB} = (r_B \cdot l)(hKT_{A1}).$$

8.4.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 2.

b) The mechanism provides mutual forward secrecy.

8.5 Key agreement of Diffie-Hellman type with 2 key pairs (KADH2KP)

This key agreement mechanism establishes a shared secret between entities *A* and *B* in two passes.

8.5.1 Setup

Prior to the process of agreeing upon a shared secret, in addition to the common information, the following must be established:

— for each entity *X*, a private key-agreement key d_x and a public key-agreement key P_x, which is an elliptic curve point satisfying $P_x = d_x G$. See Part 1 of this standard for a description of how to generate this key pair.

— for each entity, access to an authentic copy of the public key-agreement key of the other party.

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

8.5.2 Mechanism

8.5.2.1 Key token construction (A): Entity *A* randomly and secretly generates r_A in the range {1,...,n-1}, computes $r_A G$, constructs the key token,

$$KT_{A1} = r_A G$$

and sends it to *B*.

8.5.2.2 Key token construction (B): Entity *B* randomly and secretly generates r_B in the range {1,...,n-1}, computes $r_B G$, constructs the key token,

$$KT_{B1} = r_B G$$

and sends it to *A*.

8.5.2.3 Key construction (A): Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *A* computes the shared key

$$K_{AB} = (d_A \cdot l)(hKT_{B1}) \,||\, (r_A \cdot l)(hP_B).$$

8.5.2.4 Key construction (B): Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *B* computes the shared secret

$$K_{AB} = (r_B \cdot l)(hP_A) \,||\, (d_B \cdot l)(hKT_{A1}).$$

NOTE Concatenation of a representation of the points is not the only alternative for the construction of the key. Any prefix free representation (such as ASN.1) will also work. As there are choices, the method to combine the two values becomes part of what is needed to be agreed upon by all parties. See also Annex A for further discussion.

8.5.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 2.

b) The mechanism provides forward secrecy with respect to both *A* and *B* individually.

c) The mechanism provides mutual implicit key authentication.

8.6 Key agreement of Diffie-Hellman type with 2 signatures and key confirmation (KADH2SKC)

This key agreement mechanism establishes a shared secret between entities *A* and *B* in three passes.

8.6.1 Setup

Prior to the process of agreeing upon a shared secret, in addition to the common information, the following must be established:

— for each entity, a private signature key and a public verification key corresponding to a mutually agreed upon signature algorithm.

— for each entity, access to an authentic copy of the public verification key of the other party.

— any parameters to be used in the signature transformations.

— a cryptographic check function, f.

8.6.2 Mechanism

Let *X*'s private and public signature transformations be denoted S_x and V_x respectively; (S_x, V_x) could denote any signature system, for example one of the signature systems defined in ISO/IEC 9796, ISO/IEC 14888 or ISO/IEC 15946-2.

8.6.2.1 Key token construction (A): Entity *A* randomly and secretly generates r_A in the range {1,...,n-1}, computes r_AG, constructs the key token,

$$KT_{A1} = r_AG$$

and sends it to *B*.

8.6.2.2 Key token construction (B): Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *B* randomly and secretly generates r_B in the range {1,...,n-1}, computes r_BG, computes the shared secret as

$$K_{AB} = (r_B \cdot l)(hKT_{A1})$$

constructs the signed key token,

$$KT_{B1} = S_B(DB_1) \,||\, f_{K_{AB}}(DB_1)$$

where

$$DB_1 = r_B G \,||\, KT_{A1} \,||\, A \,[||\, Text1]$$

and sends it to *A*.

NOTE As a way to reduce the amount of data transmitted, if a signature scheme with appendix is used, the redundant value KT_{A1} need not be returned with the block KT_{B1}, although it still must be included within the scope of the signature calculation.

8.6.2.3 Key token processing (A): Entity *A* verifies *B*'s signature on the key token KT_{B1} using *B*'s public verification key. If a signature scheme with message recovery is used, this includes recovering the data block DB_1 from the signature and verifying that *A*'s distinguishing identifier and the value $r_A G$ are contained in it. If a signature scheme with appendix is used, this includes reconstructing the data block DB_1 using the value in KT_{A1}, *A*'s distinguishing identifier and the received value $r_B G$ and verifying the signature on that data block.

Entity *A* should verify that the value $r_B G$ obtained from KT_{B1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. If the checks are successful, *A* computes the shared key

$$K_{AB} = (r_A \cdot I)(h \cdot r_B G).$$

Using K_{AB}, *A* verifies the cryptographic check value $f_{K_{AB}}(DB_1)$. Then *A* constructs the signed key token

$$KT_{A2} = S_A(DB_2) \,||\, f_{K_{AB}}(DB_2)$$

where

$$DB_2 = r_A G \,||\, r_B G \,||\, B \,[||\, Text2]$$

and sends it to *B*.

NOTE As a way to reduce the amount of data transmitted, if a signature scheme with appendix is used, the redundant values $r_A G$ and $r_B G$ need not be returned with the block KT_{A2}, although they still must be included within the scope of the signature calculation.

8.6.2.4 Key token processing (B): Entity *B* verifies *A*'s signature on the key token KT_{A2} using *A*'s public verification key. If a signature scheme with message recovery is used, this includes recovering the data block DB_2 from the signature and verifying that *B*'s distinguishing identifier and the values $r_A G$ and $r_B G$ are contained in it. If a signature scheme with appendix is used, this includes reconstructing the data block DB_2 using the values in KT_{A1} and KT_{B1} and *B*'s distinguishing identifier and verifying the signature on that data block.

If the checks are successful *B* verifies the cryptographic check value $f_{K_{AB}}(DB_2)$ using the shared key

$$K_{AB} = (r_B \cdot I)(hKT_{A1}).$$

8.6.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 3.

b) The mechanism provides mutual forward secrecy.

c) The mechanism provides mutual explicit key authentication and mutual entity authentication.

9 Key agreement mechanisms not included in ISO/IEC 11770-3.

The key agreement mechanisms presented in this section are not included in the key management standard ISO/IEC 11770-3.

Where authentic copies of the other party's public keys are required, they can be obtained using the techniques specified in ISO/IEC 11770-3, Clause 9.

NOTE All computations in this Clause must be performed in the order given by the formulae.

9.1 Common information

For all key agreement mechanisms, prior to the process of agreeing upon a shared secret, the following common information must be established between the parties and optionally validated (see Part 1 of this standard for a description of parameter validation):

— the elliptic curve parameters with which the key pairs shall be associated, which shall be the same for both parties' key pairs. This includes p, p^m or 2^m, a description of $\boldsymbol{F}(p)$, $\boldsymbol{F}(p^m)$ or $\boldsymbol{F}(2^m)$, and an indication of the basis used, $\boldsymbol{E}$, n and G.

In each of the mechanisms defined below, the resulting agreed key should not be used as a cryptographic key directly. Instead, it should be used as the input to a key derivation function, allowing both parties to derive the same cryptographic keys from it. Hence, it is also necessary for the two parties to agree on the following information:

— a key derivation function, *kdf*;
— any parameters to the key derivation function, and
— the type of cofactor multiplication that is to be performed (if any).

9.2 The Full Unified Model

This key agreement mechanism establishes a shared secret between entities A and B in two passes.

9.2.1 Setup

Prior to the process of agreeing upon a shared secret, in addition to the common information, the following must be established:

— for each entity X, a private key-agreement key d_x, which is an integer in the range {1,...,n-1}, and a public key-agreement key P_x, which is an elliptic curve point satisfying $P_x=d_xG$.

— for each entity, access to an authenticated copy of the public key-agreement key of the other party.

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

9.2.2 Mechanism

9.2.2.1 Key Token Construction (A): Entity A randomly and secretly generates r_A in the range {1,...,n-1}, computes r_AG, constructs the key token,

$$KT_{A1} = r_AG$$

and sends it to B.

9.2.2.2 Key Token Construction (B): Entity B randomly and secretly generates r_B in the range {1,...,n-1}, computes r_BG, constructs the key token,

$$KT_{B1} = r_BG$$

and sends it to *A*.

9.2.2.3 Key Construction (A): Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *A* computes the shared key

$$K_{AB} = (r_A \cdot l)(hKT_{B1}) \,||\, (d_A \cdot l)(hP_B).$$

9.2.2.4 Key Construction (B): Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *B* computes the shared key

$$K_{AB} = (r_B \cdot l)(hKT_{A1}) \,||\, (d_B \cdot l)(hP_A).$$

9.2.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 2.

b) The mechanism provides mutual forward secrecy.

c) The mechanism provides mutual implicit key authentication.

NOTE Concatenation of a representation of the points is not the only alternative for the construction of the key. Any prefix free encoding (such as ASN.1) will also work. As there are choices, the method to combine the two values becomes part of what is needed to be agreed upon by all parties. See also Annex A for further discussion.

9.3 Key agreement of MQV type with 1 pass (KAMQV1P)

This key agreement mechanism establishes a shared secret between two entities *A* and *B* in one pass.

9.3.1 Setup

Prior to the process of agreeing upon a shared secret, in addition to the common information, the following must be established:

— for each entity X, a private key-agreement key d_x and a public key-agreement key P_x, which is an elliptic curve point satisfying $P_x = d_x G$. See Part 1 of this standard for a description of how to generate this key pair.

— for each entity, access to an authentic copy of the public key-agreement key of the other entity.

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

9.3.2 Mechanism

9.3.2.1 Key token construction (A): Entity *A* randomly and secretly generates *r* in the range {1,...,n-1}, computes *rG*, constructs the key token,

$$KT_{A1} = rG$$

and sends it to *B*.

9.3.2.2 Key construction (A): Entity *A* computes the shared key

$$K_{AB} = ((r+\pi^{*}(KT_{A1})d_A)\cdot l)(h\cdot (P_B+\pi^{*}(P_B)P_B)).$$

9.3.2.3 Key construction (B): Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Using its own private key, entity *B* computes the shared key from KT_{A1} as follows:

$$K_{AB} = ((d_B+\pi^{*}(P_B)d_B)\cdot l)(h\cdot (KT_{A1}+\pi^{*}(KT_{A1})P_A)).$$

9.3.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 1.

b) The mechanism provides mutual implicit key authentication.

c) The mechanism provides forward secrecy with respect to A.

9.4 Key agreement of MQV type with 2 passes (KAMQV2P)

This key agreement mechanism establishes a shared secret between entities *A* and *B* in two passes.

9.4.1 Setup

Prior to the process of agreeing upon a shared secret, in addition to the common information, the following must be established:

— for each entity X, a private key-agreement key d_x and a public key-agreement key P_x, which is an elliptic curve point satisfying $P_x=d_xG$. See Part 1 of this standard for a description of how to generate this key pair.

— for each entity, access to an authentic copy of the public key-agreement key of the other party.

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

9.4.2 Mechanism

9.4.2.1 Key token construction (A): Entity *A* randomly and secretly generates r_A in the range {1,...,n-1}, computes r_AG, constructs the key token,

$$KT_{A1} = r_AG$$

and sends it to *B*.

9.4.2.2 Key token construction (B): Entity *B* randomly and secretly generates r_B in the range {1,...,n-1}, computes r_BG, constructs the key token,

$$KT_{B1} = r_BG$$

and sends it to *A*.

9.4.2.3 Key construction (A): Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *A* computes the shared key

$$K_{AB} = ((r_A+\pi^{*}(KT_{A1})d_A)\cdot l)(h\cdot (KT_{B1}+\pi^{*}(KT_{B1})P_B)).$$

9.4.2.4 Key construction (B): Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. Entity *B* computes the shared key

$$K_{AB} = ((r_B+\pi^{*}(KT_{B1})d_B)\cdot l)(h\cdot (KT_{A1}+\pi^{*}(KT_{A1})P_A)).$$

9.4.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 2.

b) The mechanism provides mutual implicit key authentication.

c) The mechanism provides mutual forward secrecy.

10 Key transport mechanisms

In this part of ISO/IEC 15946 secret key transport is defined as the process of transferring a secret key, chosen by one entity (or a trusted centre), to another entity, suitably protected by elliptic curve cryptographic techniques.

NOTE All computations in this Clause must be performed in the order given by the formulae.

10.1 Common information

Prior to the process of transporting a secret key, the following common information must be established between the parties and optionally validated (see Part 1 of this standard for a description of parameter validation):

— the elliptic curve parameters with which the key pairs shall be associated, which shall be the same for both parties' key pairs, this includes p, p^m or 2^m, a description of $\boldsymbol{F}(p)$, $\boldsymbol{F}(p^m)$ or $\boldsymbol{F}(2^m)$ and an indication of the basis used, $\boldsymbol{E}$, n and G;

— a key derivation function, *kdf*;

— any parameters to the key derivation function, denoted by *parameters*,

— the type of cofactor multiplication that is to be performed (if any).

10.2 Key transport of ElGamal type (KTEG)

This key transport mechanism transfers a secret key K from entity A to entity B in one pass.

10.2.1 Setup

Prior to the process of transporting a secret key, in addition to the common information, the following must be established:

— for entity B, a private key-transport key d_B and a public key-transport key P_B, which is an elliptic curve point satisfying $P_B=d_BG$. See Part 1 of this standard for a description of how to generate this key pair;

— for entity A, access to an authentic copy of the public key-transport key of entity B;

— a MAC algorithm *MAC*;

— parameters for the key derivation function when used to produce a MAC key, *MACparameters.*

Entity A should verify that entity B's public key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

10.2.2 Mechanism

10.2.2.1 Key token construction (A): *A* randomly and secretly generates an integer *r* in the range {1,...,n-1}, computes the curve point *rG* and enciphers *K* as,

$$BE=(A||K) \text{ XOR } kdf(\pi((r\cdot l)(hP_B)), parameters).$$

Then *A* computes the intermediate key,

$$K'=kdf(\pi((r\cdot l)(hP_B)),MACparameters),$$

constructs the key token,

$$KT_{A1} = BE \,||\, rG \,||\, MAC(K',BE)$$

and sends it to *B*.

10.2.2.2 Key token deconstruction (B): Entity *B* should verify that the value *rG* obtained from KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this. To recover the key *K,* entity *B* uses its private key-transport key d_B and *rG* to determine the curve point $(d_B\cdot l)(h\cdot rG)$. *B* then calculates the projection $\pi((d_B\cdot l)(h\cdot rG))$.

B obtains the key *K* by computing

$$A||K = BE \text{ XOR } kdf(\pi((d_B\cdot l)(h\cdot rG)), parameters).$$

Finally *B* computes the intermediate key,

$$K''=kdf(\pi((d_B\cdot l)(h\cdot rG)),MACparameters),$$

verifies that the received value *MAC(K',BE)* is the same as the computed value *MAC(K'',BE)* and that the sender identification *A* is the same as what was recovered from *BE*. If all checks are successful, *B* accepts the key *K*.

10.2.3 Properties

This key transport mechanism has the following properties:

a) Number of passes: 1.

b) The mechanism provides implicit key authentication from *B* to *A* since only *B* can possibly recover the key *K*.

c) The mechanism provides forward secrecy with respect to A.

10.3 Key transport of ElGamal type with originator signature (KTEGOS)

This key transport mechanism transfers a secret key *K* from entity *A* to entity *B* in one pass.

10.3.1 Setup

Prior to the process of transporting a secret key, in addition to the common information, the following must be established:

— for entity *B*, a private key-transport key d_B and a public key-agreement key P_B, which is an elliptic curve point satisfying $P_B=d_BG$. See Part 1 of this standard for a description of how to generate this key pair;

— for entity A, access to an authentic copy of the public key-transport key of entity B;

— for entity A, a private signature key and public verification key corresponding to a mutually agreed upon signature algorithm;

— for entity B, access to an authentic copy of the public verification key of A;

— any parameters to be used in the signature transformations.

Entity *A* should verify that entity *B*'s public key-transport key is indeed a point on the elliptic curve. See Part 1 of this standard for a description of how to do this.

10.3.2 Mechanism

Let *A*'s private and public signature transformations be denoted S_A and V_A respectively; (S_A, V_A) could denote any signature system, for example one of the signature systems defined in ISO/IEC 9796, ISO/IEC 14888 or ISO/IEC 15946-2.

10.3.2.1 Key encipherment (A): Entity *A* randomly and secretly generates an integer *r* in the range {1,...,n-1} and the curve point *rG* and enciphers the key data block A||K as

$$BE=(A||K) \text{ XOR } kdf(\pi((r \cdot l)(hP_B)), parameters).$$

10.3.2.2 Key token construction (A): Entity *A* forms the key token, consisting of the recipient's distinguished identifier *B*, an optional time stamp or sequence number, *TVP*, the curve point *rG* and the enciphered block *BE*. Thus,

$$KT_{A1} = B \,||\, TVP \,||\, rG \,||\, BE$$

Then *A* signs the key token and sends the token and the resulting signature,

$$S_A(KT_{A1})$$

to *B*.

10.3.2.3 Key token verification (B) *B* uses the sender's public verification transformation V_A to verify the digital signature of the received key token KT_{A1}. Then *B* checks the receiver identification *B* and optionally the *TVP*. Entity *B* should verify that the value *rG* obtained from KT_{A1} is indeed a point on the elliptic curve. See Part 1 for a description of how to do this.

10.3.2.4 Key decipherment (B): *B* deciphers the block *BE* using its private key-transport key d_B, computing

$$A||K = BE \text{ XOR } kdf(\pi((d_B \cdot l)(h \cdot rG)), parameters).$$

Then *B* checks the sender identification *A*. If all checks are successful *B* accepts the key *K*.

10.3.3 Properties

This key agreement mechanism has the following properties:

a) Number of passes: 1.

b) The mechanism provides mutual implicit key authentication.

c) The method provides explicit key authentication from *A* to *B*.

d) The method provides forward secrecy with respect to A.

11 Key Confirmation

Explicit key confirmation is the process of adding additional messages to a key establishment protocol providing implicit key authentication so that explicit key authentication and entity authentication are provided. Explicit key confirmation can be added to any method that does not possess it inherently (e.g., Clauses 8.2, 8.3, 8.4, 9.2, 9.3 and 10.2). Key confirmation is typically provided by exchanging a value that can (in all likelihood) only be calculated correctly if the key establishment calculations were successful. Key confirmation from *A* to *B* is provided by *A* calculating a value and sending it to *B* for confirmation by *B* of *A*'s correct calculation. If mutual key confirmation is desired, then each party sends a different value to the other.

Key confirmation is often provided by immediate use of an established key, if something is wrong then it is immediately detected; this is called implicit key confirmation, explicit key confirmation in this case may be unnecessary. If one party is not online (for example, in one-pass protocols used in store and forward (email) scenarios) then it is simply not possible for the other party to obtain key confirmation. However, sometimes a key is established yet used only later (if at all) or perhaps the key establishment process may simply not know if the resulting key will be used immediately or not. In these cases, it is often desirable to use a method of explicit key confirmation, as it may otherwise be too late to correct an error once detected. Explicit key confirmation can also be seen as a way of "firming up" some security properties during the key establishment process and may be warranted if following a process of conservative protocol design.

A method of providing key confirmation using a MAC calculation is as follows:

Party *A* is the initiator, party *B* does normal key establishment processing and derives a (supposedly) shared secret *MacKey* and uses it to MAC the *MacData1* message.

1. Party *B* forms the *MacData1* octet string consisting of the message identifier octet 0x022, *B*'s identifier, *A*'s identifier, the octet string KT_B corresponding to *B*'s key token (or zeros if such does not exist), the octet string KT_A corresponding to *A*'s key token (or zeros if such does not exist), the octet string P_B corresponding to *B*'s public key-establishment key (or zeros if such does not exist), the octet string P_A corresponding to *A*'s public key-establishment key (or zeros if such does not exist) and if present optional additional *Text1*:

   $$MacData1 = 02 \,||\, B \,||\, A \,||\, KT_B \,||\, KT_A \,||\, P_B \,||\, P_A \,[||\, Text1]. \text{ Where 0x02 is the msg. number.}$$

2. Party B calculates

   $$MacKey_B = kdf(K_{AB}).$$

   Party *B* calculates *MacTag1* for *MacData1* under the (supposedly) shared secret $MacKey_B$ of an appropriate MAC scheme specified.

3. Party *B* sends *MacData1*, *MacTag1* to party *A*.

4. Party *A* calculates

   $$MacKey_A = kdf(K_{AB})$$

   and verifies *MacTag1* on the *MacData1* message.

5. Assuming the MAC verifies, party *A* has received key confirmation from party *B* (that is, party *A* knows that $MacKey_A$ equals $MacKey_B$). If mutual key confirmation is desired, party *A* continues the protocol and forms the *MacData2* message. Form the octet string consisting of the message identifier octet 0x03, *A*'s identifier, *B*'s identifier, the octet string KT_A corresponding to *A*'s key token (or zeros if such does not exist), the octet string KT_B corresponding to *B*'s key token (or zeros if such does not exist), the octet string P_A corresponding to *A*'s public key-establishment key (or zeros if such does not exist), the octet string P_B corresponding to *B*'s public key-establishment key (or zeros if such does not exist) and optional additional octet string *Text2*:

   $$MacData2 = 03 \,||\, A \,||\, B \,||\, KT_A \,||\, KT_B \,||\, P_A \,||\, P_B \,[||Text2] \text{ Where 0x03 is the msg. number.}$$

6. Party *A* calculates the tag *MacTag2* on *MacData2* under the (supposedly) shared secret $MacKey_A$ using an appropriate MAC scheme.

7. Party *A* sends *MacData2*, *MacTag2* to Party *B*.

8. Party *B* uses $MacKey_B$ to verify the MAC on the *MacData2* message. Assuming the MAC verifies, party *B* has received key confirmation from party *A* (that is, party *B* knows that $MacKey_A$ equals $MacKey_B$).

Other methods of key confirmation are possible. If the shared secret is to be used for data confidentiality (encryption), one party can send the encryption of some specific plaintext known to the other party, for example, a block of all binary zeros or all binary ones. Care should be taken that any subsequent use of the key is very unlikely to encrypt the same plaintext as was used for key confirmation.

Annex A
(informative)

Examples of key derivation functions

A.1 The IEEE P1363 key derivation function

This clause describes the key derivation function that is given in the IEEE P1363 standard [3].

A.1.1 Preconditions

As a precondition of the use of this key derivation function, users must agree on a common hash function *hash*. Users who use different hash functions will obtain different results. For the purposes of this standard, the hash function must be one described in ISO/IEC 10118. The shared key that is produced will have length equal to the length of the output of the hash function.

A.1.2 Input

The inputs to this key derivation function are

1. The shared secret *z* which is an integer, expressed as an octet string.

   NOTE The mechanisms in Clauses 8, 9 and 10 derive shared keys K_{AB} either as points on the elliptic curve or as the concatenation of two points on an elliptic curve. In the first situation, in order to obtain a shared secret integer *z* for input into the key derivation function, the function π should be applied to the point. In the second situation, the function π should be applied to both points to obtain two integers z_1 and z_2. The two integers should then be converted to octet strings and concatenated (or combined using any prefix-free encoding method), as were the points, to obtain the appropriate octet string.

2. The key derivation parameters, *parameters*, also expressed as an octet string.

   NOTE Users must also agree on a common method of converting integers and parameters to octet strings for input into the key derivation function.

A.1.3 Actions

If the combined length of the shared secret *z* and the parameters exceeds any limitation that may exist for the agreed hash function, *hash*, then output “error” and stop.

Otherwise compute the value *K=hash(z||parameters)*.

A.1.4 Output

Output *K* as the shared secret.

A.2 The ANSI X9.42 key derivation function

This clause describes a key derivation function based on the key derivation function that is given in the ANSI X9.42 standard [1].

A.2.1 Prerequisites

A hash function *hash* specified in ISO/IEC 10118 must be chosen. Let *hashlen* denote the length of the output of the hash function chosen, and let *maxhashlen* denote the maximum length of the input to the hash function.

A.2.2 Input

The input to the key derivation function is:

1. *ZZ*: A bit string denoting the shared secret.

   NOTE The mechanisms in Clauses 8, 9 and 10 derive shared keys K_{AB} either as points on the elliptic curve or as the concatenation of two points on an elliptic curve. In the first situation, in order to obtain a shared secret value *ZZ* for input into the key derivation function, the function π should be applied to the point and the resulting integer converted to a bit string. In the second situation, the function π should be applied to both points to obtain two integers z1 and z2. The two integers should then be converted to bit strings and concatenated (or combined using any prefix-free encoding method), as were the points, to obtain the appropriate bit string.

2. *keydatalen*: An integer representing the length in bits of the keying data to be generated. This integer is less than $(hashlen \times (2^{32}-1))$.

3. *OtherInfo*: A bit string, specified in ASN.1 DER encoding, consisting of the following information as specified in Clause A.2.5.

   3.1. Key specification information consisting of :

      3.1.1. *AlgorithmID*: a unique object identifier (OID) of the symmetric algorithm(s) with which the keying data will be used.

      3.1.2. *Counter*: a 32-bit octet string, with initial value 00000001_{16}.

   3.2. (Optional) *PartyUInfo*: A bit string containing public information contributed by the initiator.

   3.3. (Optional) *PartyVInfo*: A bit string containing public information contributed by the responder.

   3.4. (Optional) *SuppPrivInfo*: A bit string containing some additional, mutually known private information, e.g. a shared secret symmetric key communicated through a separate channel.

   3.5. (Optional) *SuppPubInfo*: A bit string containing some additional, mutually known public information.

   NOTE Users must also agree on a common method of converting integers and parameters to bit strings for input into the key derivation function.

A.2.3 Actions

The key derivation function is computed as follows:

1. Let $d = \lceil keydatalen / hashlen \rceil$.

2. Initialise *Counter* = 00000001_{16}.

3. For $i = 1$ to d,

   3.1. Compute $h_i = hash(ZZ||OtherInfo)$ where h_i denotes the hash value computed using the appropriate hash function.

   3.2. Increment *Counter*.

   3.3. Increment *i*.

4. Compute K = leftmost *keydatalen* bits of $h_1||h_2||\ldots||h_d$.

5. Output K.

A.2.4 Output

The keying data *K* as a bit string of length *keydatalen* bits.

Note that this key derivation function based on ASN.1 DER encoding produces keying data which is less than *hashlen*×*(*2^{32}–1*)* bits in length. It is assumed that all key derivation function calls are indeed for bit strings which are less than *hashlen*×*(*2^{32}–1*)* bits in length. Any scheme attempting to call the key derivation function using a bit string that is greater than or equal to *hashlen*×*(*2^{32}–1*)* bits shall output "invalid" and stop. Similarly, it is assumed that all key derivation function calls do not involve hashing a bit string that is more than *maxhashlen* bits in length. Any scheme attempting to call the key derivation function on a call involving hashing a bit string that is greater than *maxhashlen* bits shall output "invalid" and stop.

A.2.5 ASN.1 syntax

The input to the key derivation function is the shared secret *ZZ* and other information *OtherInfo*.

The other information includes the initiator's information *PartyUInfo*, the responder's information *PartyVInfo*, *suppPubInfo*, and *suppPrivInfo*.

```
OtherInfo ::= SEQUENCE {
    keyInfo             KeySpecificInfo,
    partyUInfo          [0] OCTET STRING  OPTIONAL,
    partyVInfo          [1] OCTET STRING  OPTIONAL,
    suppPubInfo                 [2] OCTET STRING  OPTIONAL,
    suppPrivInfo                [3] OCTET STRING  OPTIONAL
}
KeySpecificInfo ::= SEQUENCE {
    algorithm           OBJECT IDENTIFIER,
    counter             Counter
 }

Counter ::= INTEGER (1...32767)
```

The *suppPubInfo* and *suppPrivInfo* fields are optional fields used in key derivation. These fields may be used to hold additional, supplementary public and private information that is mutually known to the communicating parties, but that is not specific to either party.

The contents of *suppPubInfo* and *suppPrivInfo* are defined by the key management protocol. The definition, syntax, and encoding rules of the *suppPubInfo* and *suppPrivInfo* fields are the responsibility of the key management protocol and are beyond the scope of this standard.

All inputs to the key derivation hash function shall be an integral number of octets in length. *suppPrivInfo* may include *ZZ*.

A.3 The ANSI X9.63 key derivation function

This clause describes a key derivation function based on the key derivation function that is given in the ANSI X9.63 standard [2].