
**Information technology — Security
techniques — Key management —**

**Part 3:
Mechanisms using asymmetric
techniques**

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

Partie 3: Mécanismes utilisant des techniques asymétriques



Reference number
ISO/IEC 11770-3:2008(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2008

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2008

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents

Page

Foreword.....	iv
Introduction	v
1 Scope	1
2 Normative references	2
3 Terms and definitions.....	2
4 Symbols and abbreviations	6
5 Requirements	8
6 Key derivation functions	8
7 Cofactor multiplication	9
8 Key commitment.....	10
9 Key confirmation.....	10
10 Secret key agreement.....	12
10.1 Key agreement mechanism 1	12
10.2 Key agreement mechanism 2	13
10.3 Key agreement mechanism 3	14
10.4 Key agreement mechanism 4	16
10.5 Key agreement mechanism 5	17
10.6 Key agreement mechanism 6	18
10.7 Key agreement mechanism 7	20
10.8 Key agreement mechanism 8	22
10.9 Key agreement mechanism 9	23
10.10 Key agreement mechanism 10	24
10.11 Key agreement mechanism 11	25
11 Secret key transport	27
11.1 Key transport mechanism 1.....	27
11.2 Key transport mechanism 2.....	28
11.3 Key transport mechanism 3.....	30
11.4 Key transport mechanism 4.....	31
11.5 Key transport mechanism 5.....	33
11.6 Key transport mechanism 6.....	35
12 Public key transport	37
12.1 Public key distribution without a trusted third party	37
12.2 Public key distribution using a trusted third party	39
Annex A (informative) Properties of key establishment mechanisms.....	41
Annex B (informative) Examples of key derivation functions	43
Annex C (informative) Examples of key establishment mechanisms.....	51
Annex D (informative) Examples of elliptic curve based key establishment mechanisms	57
Annex E (informative) Key transport	69
Annex F (normative) ASN.1 module	74
Bibliography	82

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

ISO/IEC 11770-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *IT Security techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 11770-3:1999), and ISO/IEC 15946-3:2002, which have been merged and updated to present a uniform standard on key management.

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

- *Part 1: Framework*
- *Part 2: Mechanisms using symmetric techniques*
- *Part 3: Mechanisms using asymmetric techniques*
- *Part 4: Mechanisms based on weak secrets*

Introduction

This part of ISO/IEC 11770 defines schemes that can be used for key agreement and schemes that can be used for key transport.

Public key cryptosystems were first proposed in the seminal paper by Diffie and Hellman in 1976. The security of many cryptosystems is based on the presumed intractability of solving the discrete logarithm problem over a finite field. Other cryptosystems such as RSA are based on the difficulty of the integer factorization problem.

The second form of cryptography discussed in this part of ISO/IEC 11770 is based on elliptic curves. 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 V. Miller and N. 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.

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:

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

SD 8 is publicly available at <http://www.jtc1sc27.din.de/sce/sd8>.

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.

Information technology — Security techniques — Key management —

Part 3: Mechanisms using asymmetric techniques

1 Scope

This part of ISO/IEC 11770 defines key management mechanisms based on asymmetric cryptographic techniques. It specifically addresses the use of asymmetric techniques to achieve the following goals.

- 1) Establish a shared secret key for a symmetric cryptographic technique between two entities *A* and *B* by key agreement. In a secret key agreement mechanism, the secret key is the result of a data exchange between the two entities *A* and *B*. Neither of them can predetermine the value of the shared secret key.
- 2) Establish a shared secret key for a symmetric cryptographic technique between two entities *A* and *B* by key transport. In a secret key transport mechanism, the secret key is chosen by one entity *A* and is transferred to another entity *B*, suitably protected by asymmetric techniques.
- 3) Make an entity's public key available to other entities by key transport. In a public key transport mechanism, the public key of entity *A* must be transferred to other entities in an authenticated way, but not requiring secrecy.

Some of the mechanisms of this part of ISO/IEC 11770 are based on the corresponding authentication mechanisms in ISO/IEC 9798-3.

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

- key lifecycle management,
- mechanisms to generate or validate asymmetric key pairs,
- mechanisms to store, archive, delete, destroy, etc. keys.

While this part of ISO/IEC 11770 does not explicitly cover the distribution of an entity's private key (of an asymmetric key pair) from a trusted third party to a requesting entity, the key transport mechanisms described can be used to achieve this. A private key can in all cases be distributed with these mechanisms where an existing, non-compromised key already exists. However, in practice the distribution of private keys is usually a manual process that relies on technological means like smart cards, etc.

This part of ISO/IEC 11770 does not cover the implementations of the transformations used in the key management mechanisms.

NOTE To achieve authenticity of key management messages, it is possible to make provisions for authenticity within the key establishment protocol or to use a public key signature system to sign the key exchange messages.

2 Normative references

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

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

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

3 Terms and definitions

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

3.1

asymmetric cryptographic technique

cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key), and has the property that given the public transformation, it is computationally infeasible to derive the private transformation

NOTE A system based on asymmetric cryptographic techniques can either be an encipherment system, a signature system, a combined encipherment and signature system, or a key agreement system. With asymmetric cryptographic techniques there are four elementary transformations: sign and verify for signature systems, encipher and decipher for encipherment systems. The signature and the decipherment transformation are kept private by the owning entity, whereas the corresponding verification and encipherment transformations are published. There exist asymmetric cryptosystems (e.g. RSA) where the four elementary functions can be achieved by only two transformations: one private transformation suffices for both signing and decrypting messages, and one public transformation suffices for both verifying and encrypting messages. However, since this does not conform to the principle of key separation, throughout this part of ISO/IEC 11770, the four elementary transformations and the corresponding keys are kept separate.

3.2

asymmetric encipherment system

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

pair of related keys where the private key defines the private transformation and the public key defines the public transformation

3.4

certification authority

CA

centre trusted to create and assign public key certificates

3.5

decipherment

reversal of a corresponding encipherment

[ISO/IEC 11770-1:1996]

3.6

digital signature

data unit appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the origin and integrity of the data unit and protect the sender and the recipient of the data unit against forgery by third parties, and the sender against forgery by the recipient

3.7**distinguishing identifier**

information which unambiguously distinguishes an entity

[ISO/IEC 11770-1:1996]

3.8**encipherment**

(reversible) transformation of data by a cryptographic algorithm to produce ciphertext, i.e. to hide the information content of the data

[ISO/IEC 9797-1:1999, ISO/IEC 11770-1:1996, ISO/IEC 18033-1:2005]

3.9**entity authentication**

corroboration that an entity is the one claimed

[ISO/IEC 9798-1:1997]

3.10**entity authentication of entity A to entity B**

assurance of the identity of entity A for entity B

3.11**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 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.

3.12**forward secrecy with respect to entity A**

property that knowledge of entity A's long-term private key subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys

3.13**forward secrecy with respect to both entity A and entity B individually**

property that knowledge of entity A's long-term private key or knowledge of entity 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 entity A's and entity B's long-term private keys do not enable recomputation of previously derived keys.

3.14**hash-function**

function which maps strings of bits to fixed-length strings of bits, satisfying two properties:

- 1) it is computationally infeasible to find for a given output, an input which maps to this output;
- 2) it is computationally infeasible to find for a given input, a second input which maps to the same output

NOTE 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.

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

3.15**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

3.16

key

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

[ISO/IEC 11770-1:1996]

3.17

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 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.

3.18

key commitment

process of committing to use specific keys in the operation of a key agreement scheme before revealing the specified keys

3.19

key confirmation from entity *A* to entity *B*

assurance for entity *B* that entity *A* is in possession of the correct key

3.20

key control

ability to choose the key or the parameters used in the key computation

3.21

key derivation function

function that outputs one or more shared secrets, used as keys, given shared secrets and other mutually known parameters as input

3.22

key establishment

process of making available a shared secret key to one or more entities, where the process includes key agreement and key transport

3.23

key token

key management message sent from one entity to another entity during the execution of a key management mechanism

3.24

key transport

process of transferring a key from one entity to another entity, suitably protected

3.25

message authentication code

MAC

string of bits which is the output of a *MAC* algorithm

NOTE A *MAC* is sometimes called a cryptographic check value (see, for example, ISO 7498-2).

3.26

message authentication code (MAC) 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:

- 1) for any key and any input string the function can be computed efficiently;

2) 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 the 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

3.27

mutual entity authentication

entity authentication which provides both entities with assurance of each other's identity

3.28

mutual forward secrecy

property that knowledge of both entity A 's and entity B 's long-term private keys subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys

3.29

one-way function

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

3.30

prefix free representation

representation of a data element for which concatenation with any other data does not produce a valid representation

3.31

private key

key of an entity's asymmetric key pair which can 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.

3.32

public key

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 can be available only to all members of a pre-specified group.

3.33

public key certificate

public key information of an entity signed by the certification authority and thereby rendered unforgeable

3.34

public key information

information containing at least the entity's distinguishing identifier and public key, but which can include other static information regarding the certification authority, the entity, restrictions on key usage, the validity period, or the involved algorithms

3.35

secret key

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

3.36

sequence number

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]

3.37

signature system

system based on asymmetric cryptographic techniques whose private transformation is used for signing and whose public transformation is used for verification

3.38

time stamp

data item which denotes a point in time with respect to a common time reference

3.39

time stamping authority

trusted third party trusted to provide evidence which includes the time when the secure time stamp is generated

[ISO/IEC 13888-1:2004]

3.40

time variant parameter

TVP

data item used to verify that a message is not a replay, such as a random number, a sequence number, or a time stamp

NOTE If time stamps are used, secure and synchronized time clocks are required. If sequence numbers are used, the ability to maintain and verify bilateral counters is required.

3.41

trusted third party

security authority, or its agent, trusted by other entities with respect to security related activities

[ISO/IEC 10181-1:1996]

4 Symbols and abbreviations

A, B distinguishing identifiers of entities

BE enciphered data block

BS signed data block

CA certification authority

$Cert_A$ entity A 's public key certificate

D_A entity A 's private decipherment transformation

d_A entity A 's private decipherment key

E An elliptic curve, either given by an equation of the form $Y^2 = X^3 + aX + b$ over the field $F(p^m)$ for $p > 3$, by an equation of the form $Y^2 + XY = X^3 + aX^2 + b$ over the field $F(2^m)$, or by an equation of the form $Y^2 = X^3 + aX^2 + b$ over the field $F(3^m)$, together with an extra point O_E referred to as the point at infinity. The elliptic curve is denoted by $E/F(p^m)$, $E/F(2^m)$, or $E/F(3^m)$, respectively.

E_A entity A 's public encipherment transformation

e_A entity A 's public encipherment key

F the key agreement function

$F(h,g)$	the key agreement function using as input a cofactor h and a common element g
G	a point on E with order n
g	the common element shared publicly by all the entities that use the key agreement function F
H	set of elements
$hash$	hash-function
h_A	entity A 's private key agreement key
j	the cofactor used in performing cofactor multiplication
K	a secret key for a symmetric cryptosystem
KT	key token
KT_A	entity A 's key token
KT_{Ai}	the key token sent by entity A after processing phase i
K_{AB}	a secret key shared between entities A and B
	NOTE In practical implementations, the shared secret key should be subject to further processing before it can be used for a symmetric cryptosystem.
kdf	a key derivation function
l	a supplementary value used in performing cofactor multiplication
M	a data message
MAC	Message Authentication Code
$MAC_K(Z)$	The output of a MAC algorithm when using as input the secret key K and an arbitrary data string Z
MQV	Menezes-Qu-Vanstone
n	a prime divisor of the order (or cardinality) of an elliptic curve E over a finite field
O_E	the elliptic curve point at infinity
P	a point on an elliptic curve E
P_X	the x-coordinate of a point P
PKI_A	entity A 's public key information
$parameters$	parameters used in the key derivation function
p_A	entity A 's public key agreement key
q	a prime power p^m for some prime $p \neq 3$ and some integer $m \geq 1$
r	a random number generated in the course of a mechanism
r_A	a random number issued by entity A in a key agreement mechanism

S	set of elements
S_A	entity A 's private signature transformation
s_A	entity A 's private signature key
TVP	time-variant parameter such as a random number, a time stamp, or a sequence number
$Text_i$	i th optional text, data or other information that may be included in a data block, if desired
V_A	entity A 's public verification transformation
v_A	entity A 's public verification key
w	one-way function
$\#E$	The order (or cardinality) of an elliptic curve E
$\parallel$	concatenation of two data elements
$\lceil x \rceil$	the smallest integer greater than or equal to the real number x
Σ	the digital signature
$\pi(P)$	$(P_x \bmod 2^{\lceil \rho/2 \rceil}) + 2^{\lceil \rho/2 \rceil}$ where $\rho = \lceil \log_2 n \rceil$ and P_x is the x-coordinate of the point P

NOTE 1 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 Σ itself. In the case of a signature system with appendix, $S_A(m)$ denotes the message m together with the signature Σ .

NOTE 2 The keys of an asymmetric cryptosystem are denoted by a lower case letter (indicating the function of that key) indexed with the identifier of its owner, e.g. the public verification key of entity A is denoted by v_A . The corresponding transformations are denoted by upper case letters indexed with the identifier of their owner, e.g. the public verification transformation of entity A is denoted by V_A .

5 Requirements

It is assumed that the entities are aware of each other's claimed identities. This may be achieved by the inclusion of identifiers in information exchanged between the two entities, or it may be apparent from the context of the use of the mechanism. Verifying the identity means to check that a received identifier field agrees with some known (trusted) value or prior expectation.

If a public key is registered with an entity, then that entity shall make sure that the entity who registers the key is in possession of the corresponding private key (see ISO/IEC 11770-1 for further guidance on key registration).

6 Key derivation functions

The use of a shared secret as derived in Clause 10 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 produces keys that are computationally indistinguishable from randomly generated keys. The key derivation function takes as input the shared secret and a set of key derivation parameters and produce an output of the desired length.

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 part of ISO/IEC 11770.

See Annex B for examples of key derivation functions.

7 Cofactor multiplication

This clause applies only to elliptic curve cryptography. The key agreement mechanisms in Clause 10 and the key transport mechanisms in Clauses 11 and 12 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 ISO/IEC 11770-1 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 j and l , defined below, are used for cofactor multiplication in Clause 10.

If cofactor multiplication is desired, there are two options:

- 1) If cofactor multiplication is used, and incompatibility with those not using cofactor multiplication is desired, then let $j = \#E / n$ and $l = 1$. If this option is chosen, both parties involved must agree to use this option, otherwise the mechanism will not work.
- 2) If cofactor multiplication is used, and compatibility with those not using cofactor multiplication is desired, then let $j = \#E / n$ and $l = j^{-1} \bmod n$.

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

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

- 1) If cofactor multiplication is not used, then let $j = 1$ and $l = 1$.

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 (O_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.

8 Key commitment

Clause 10 describes 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 [19].

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 entity hash the public key or key token and send the hash-code to the second entity, the second entity replies with his public key or key token and the first entity replies with his public key or key token, which the second entity can hash and verify that the result is equal to the hash-code sent earlier.

ISO 22895 defines cryptographic syntax and processing that can be used to implement the key commitment protocol defined in this part of ISO/IEC 11770. A *DigestedData* message, with its optional content absent, is used to send the key commitment hash along with the digest algorithm and any associated parameters. The same message with its optional content present is used to exchange the keys. The message recipient retains the first hash and ensures that this value is identical to the hash on the keys in the second message. When origin authentication of the key commitment is needed, the *DigestedData* message can be encapsulated in a *SignedData* message. These ISO 22895 messages can be represented in a compact binary form or as XML markup.

9 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. 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 entity *A* to entity *B* is provided by entity *A* calculating a value and sending it to entity *B* for confirmation by entity *B* of entity *A*'s correct calculation. If mutual key confirmation is desired, then each entity sends a different value to the other.

Key confirmation is often provided by subsequent use of an established key, and 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 entity 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 entity 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.

An example method of providing key confirmation using a *MAC* is as follows:

Entities *A* and *B* first perform one of the key establishment procedures specified in Clauses 10 and 11 of this part of ISO/IEC 11770. As a result, they expect to share a secret *MAC* key K_{AB} . They then perform the following procedure.

- 1) Entity *B* forms the message *M* octet string consisting of the message identifier octet 0x02, entity *B*'s identifier, entity *A*'s identifier, the octet string KT_B corresponding to entity *B*'s key token (or omit the fields if such does not exist), the octet string KT_A corresponding to entity *A*'s key token (or omit the fields if such does not exist), the octet string P_B corresponding to entity *B*'s public key-establishment key (or omit the fields if such does not exist), the octet string P_A corresponding to entity *A*'s public key-establishment key (or omit the fields if such does not exist) and if present optional additional *Text1*:

$$M = 02 \parallel B \parallel A \parallel KT_B \parallel KT_A \parallel P_B \parallel P_A \parallel Text1,$$

where $0x02$ is the message number.

- 2) Entity B calculates

$$K_B = \text{kdf}(K_{AB})$$

Entity B then calculates

$$\text{MAC}_{K_B}(M)$$

for the message M under the (supposedly) shared secret key K_B of an appropriate MAC scheme.

- 3) Entity B sends the message M and $\text{MAC}_{K_B}(M)$ to entity A .
 4) Entity A calculates

$$K_A = \text{kdf}(K_{AB})$$

and verifies $\text{MAC}_{K_B}(M)$ on the message M .

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

$$M' = 03 \parallel A \parallel B \parallel KT_A \parallel KT_B \parallel P_A \parallel P_B \parallel \text{Text2},$$

where $0x03$ is the message number.

- 6) Entity A calculates

$$\text{MAC}_{K_A}(M')$$

under the (supposedly) shared secret K_A using an appropriate MAC scheme.

- 7) Entity A sends M' and $\text{MAC}_{K_A}(M')$ to entity B .
 8) Entity B uses K_B to verify $\text{MAC}_{K_A}(M')$ on the message M' . Assuming the MAC verifies, entity B has received key confirmation from entity A (that is, entity B knows that K_A equals K_B).

Other methods of key confirmation are possible. If the shared secret is to be used for data confidentiality (encryption), one entity can send the encryption of some specific plaintext known to the other entity, 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. Furthermore, the MAC key should not be reused after key confirmation.

10 Secret key agreement

Key agreement is the process of establishing a shared secret key between two entities A and B in such a way that neither of them can predetermine the value of the shared secret key. Key agreement mechanisms may provide for implicit key authentication; in the context of key establishment, implicit key authentication means that after the execution of the mechanism only an identified entity can be in possession of the correct shared secret key.

The key agreement between two entities A and B takes place in a context shared by the two entities. The context consists of the following objects: a set S , a set H and a function F . The function F shall satisfy the following requirements:

- 9) F operates on two inputs, one element h from H and one element g from S , and produces a result y in S , $y = F(h, g)$.
- 10) F satisfies the commutativity condition $F(h_A, F(h_B, g)) = F(h_B, F(h_A, g))$.
- 11) It is computationally intractable to find $F(h_1, F(h_2, g))$ from $F(h_1, g)$, $F(h_2, g)$ and g . This implies that $F(\cdot, g)$ is a one-way function.
- 12) The entities A and B share a common element g in S which may be publicly known.
- 13) The entities acting on this setting can efficiently compute function values $F(h, g)$ and can efficiently generate random elements in H .

Depending on the particular key agreement mechanism further conditions may be imposed.

NOTES

- 1) Examples of a possible function F are given in Annexes C and D.
- 2) In practical implementations of the key agreement mechanisms the shared secret key should be subject to further processing. A derived shared secret key should be computed (1) by extracting bits from the shared secret key K_{AB} directly or (2) by passing the shared secret K_{AB} and optionally other nonsecret data through a one-way function and extracting bits from the output.
- 3) It will in general be necessary to check the received function values $F(h, g)$ for weak values. If such values are encountered, the protocol shall be aborted. An example known as the Diffie-Hellman key agreement is given in Annex C.5.

10.1 Key agreement mechanism 1

This key agreement mechanism non-interactively establishes a shared secret key between entities A and B with mutual implicit key authentication. The following requirements shall be satisfied:

- 1) Each entity X has a private key agreement key h_X in H and a public key agreement key $p_X = F(h_X, g)$.
- 2) Each entity has access to an authenticated copy of the public key agreement key of the other entity. This may be achieved using the mechanisms of Clause 12.

See Figure 1 for a diagram of key agreement mechanism 1.

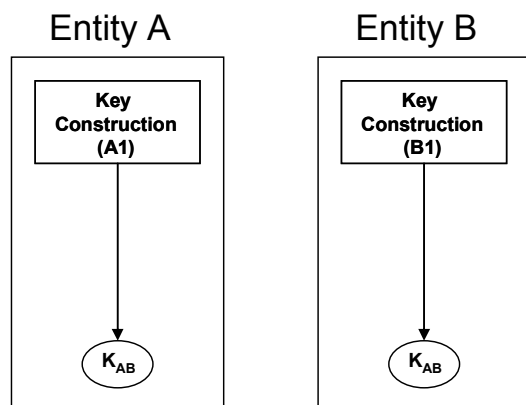


Figure 1: Key Agreement Mechanism 1

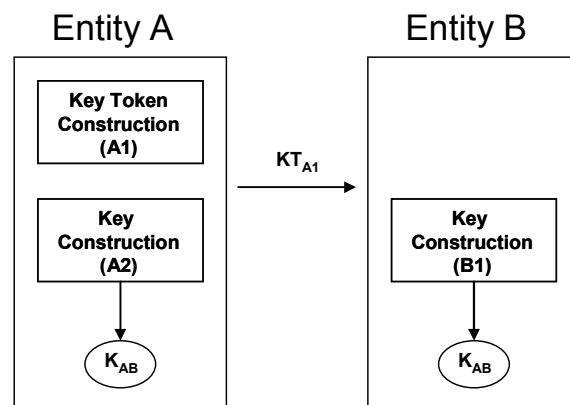


Figure 2: Key Agreement Mechanisms 2, 8

Key Construction (A1) Entity A computes, using its own private key agreement key h_A and entity B's public key agreement key p_B , the shared secret key as

$$K_{AB} = F(h_A, p_B).$$

Key Construction (B1) Entity B computes, using its own private key agreement key h_B and entity A's public key agreement key p_A , the shared secret key as

$$K_{AB} = F(h_B, p_A).$$

As a consequence of requirement 2 of F in Clause 10, the two computed values for the key K_{AB} are identical.

NOTES - This Key Agreement Mechanism has the following properties:

- 1) Number of passes: 0. As a consequence, the secret shared key has always the same value (but see Clause 10 note 2).
- 2) Key authentication: this mechanism provides mutual implicit key authentication. However, the single repeated key is a problem in a zero pass protocol. One way to eliminate this problem is to ensure that the key is only used once. Furthermore, the use of an initialization vector with small amounts of data can also solve this problem.
- 3) Key confirmation: this mechanism provides no key confirmation.
- 4) This is a key agreement mechanism since the established key is a one-way function of the private key agreement keys h_A and h_B of entities A and B respectively. However, one entity may know the other entity's public key prior to choosing their private key. Such an entity may select approximately s bits of the established key, at the cost of generating 2^s candidate values for their private key agreement key in the interval between discovering the other entity's public key and choosing their own private key.
- 5) Example: examples known as the Diffie-Hellman key agreement are given in Annexes C.1, C.2, and D.2.

10.2 Key agreement mechanism 2

This key agreement mechanism establishes in one pass a shared secret key between entities A and B with implicit key authentication from entity B to entity A, but no entity authentication from entity A to entity B (i.e. entity B does not know with whom it has established the shared secret key). The following requirements shall be satisfied:

- 1) Entity *B* has a private key agreement key h_B in H and a public key agreement key $p_B = F(h_B, g)$.
- 2) Entity *A* has access to an authenticated copy of entity *B*'s public key agreement key p_B . This may be achieved using the mechanisms of Clause 12.

See Figure 2 for a diagram of key agreement mechanism 2.

Key Token Construction (A1) Entity *A* randomly and secretly generates r in H , computes $F(r, g)$ and sends the key token

$$KT_{A1} = F(r, g) || \text{Text}$$

to entity *B*

Key Construction (A2) Entity *A* further computes the key as

$$K_{AB} = F(r, p_B).$$

Key Construction (B1) Entity *B* extracts $F(r, g)$ from the received key token KT_{A1} and computes the shared secret key

$$K_{AB} = F(h_B, F(r, g)).$$

According to requirement 2 of F in Clause 10, the two computed values for the key K_{AB} are identical.

NOTES - This Key Agreement Mechanism has the following properties:

- 1) Number of passes: 1.
- 2) Key authentication: this mechanism provides implicit key authentication from entity *B* to entity *A* (entity *B* is the only entity other than entity *A* who can compute the shared secret key).
- 3) Key confirmation: this mechanism provides no key confirmation.
- 4) This is a key agreement mechanism, since the established key is a one-way function of a random value r supplied by entity *A* and entity *B*'s private key agreement key. As discussed in Clause 8, since entity *A* could know entity *B*'s public key prior to choosing the value r , entity *A* may select approximately s bits of the established key, at the cost of generating 2^s candidate values for r in the interval between discovering entity *B*'s public key and sending KT_{A1} .
- 5) Example: examples of this key agreement mechanism (known as ElGamal key agreement) are described in Annexes C.3 and D.3.
- 6) Key usage: as entity *B* receives the necessary information to compute the key K_{AB} from the non-authenticated entity *A*, secure usage of K_{AB} at entity *B*'s end is restricted to functions not requiring trust in entity *A*'s authenticity such as decipherment and generation of message authentication codes.

10.3 Key agreement mechanism 3

This key agreement mechanism establishes in one pass a shared secret key between entities *A* and *B* with mutual implicit key authentication, and entity authentication of entity *A* to entity *B*. The following requirements shall be satisfied:

- 1) Entity *A* has an asymmetric signature system (S_A, V_A) .
- 2) Entity *B* has access to an authenticated copy of the public verification transformation V_A . This may be achieved using the mechanisms of Clause 12.

- 3) Entity *B* has a key agreement system with keys (h_B, p_B) .
- 4) Entity *A* has access to an authenticated copy of the public key agreement key p_B of entity *B*. This may be achieved using the mechanisms of Clause 12.
- 5) (Optional) The *TVP* shall either be a time stamp or a sequence number. If time stamps are used, secure and synchronized time clocks are required; if sequence numbers are used, the ability to maintain and verify bilateral counters is required.
- 6) The entities *A* and *B* have agreed on a *MAC* function (such as those standardized in ISO/IEC 9797) and a way to incorporate K_{AB} as the key in this check function.

See Figure 3 for a diagram of key agreement mechanism 3.

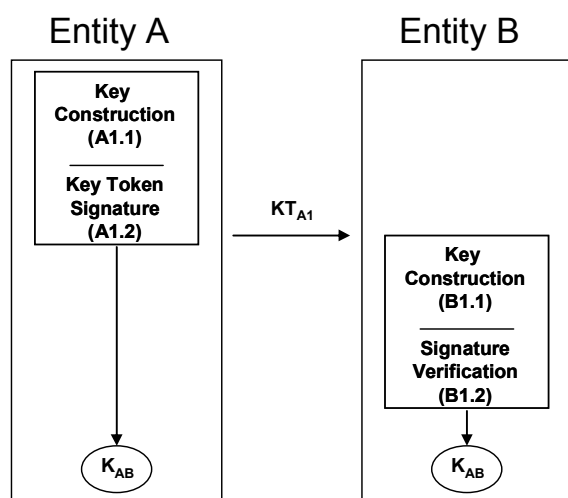


Figure 3: Key Agreement Mechanism 3

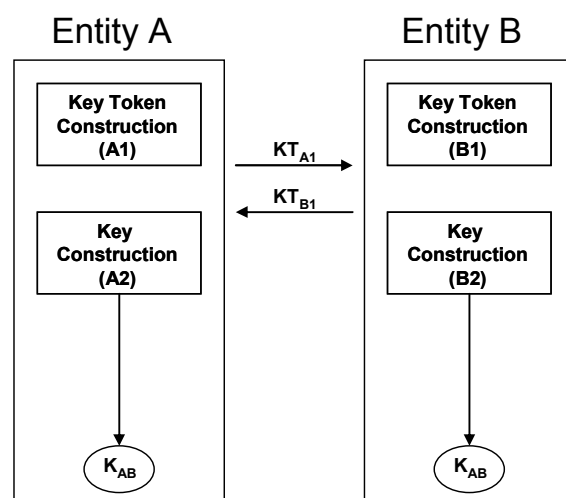


Figure 4: Key Agreement Mechanisms 4, 5, 9

Key Construction (A1.1) Entity *A* randomly and secretly generates r in H and computes $F(r, g)$. Entity *A* computes the shared secret key as

$$K_{AB} = F(r, p_B).$$

Using the shared secret key K_{AB} , entity *A* computes a *MAC* on the concatenation of the sender's distinguishing identifier for entity *A* and an optional time stamp *TVP* or sequence number.

Key Token Signature (A1.2) Entity *A* signs the *MAC*, using its private signature transformation S_A . Then entity *A* forms the key token, consisting of the sender's distinguishing identifier for entity *A*, the key input $F(r, g)$, the (optional) *TVP*, the signed *MAC*, and some optional data

$$KT_{A1} = A || F(r, g) || TVP || S_A(MAC_{K_{AB}}(A || TVP)) || \text{Text1}$$

and sends it to entity *B*.

Key Construction (B1.1) Entity *B* extracts $F(r, g)$ from the received key token and computes the shared secret key, using its private key agreement key h_B ,

$$K_{AB} = F(h_B, F(r, g)).$$

Using the shared secret key K_{AB} , entity *B* computes the *MAC* on the sender's distinguishing identifier for entity *A* and the (optional) *TVP*.

Signature Verification (B1.2) Entity *B* uses the sender's public verification transformation V_A to verify entity *A*'s signature and thus the integrity and origin of the received key token KT_{A1} . Then entity *B* validates the timeliness of the token (by inspection of the (optional) *TVP*).

NOTES - This Key Agreement Mechanism has the following properties:

- 1) Number of passes: 1.
- 2) Key authentication: this mechanism provides explicit key authentication from entity *A* to entity *B* and implicit key authentication from entity *B* to entity *A*.
- 3) Key confirmation: this mechanism provides key confirmation from entity *A* to entity *B*.
- 4) This is a key agreement mechanism, since the established key is a one-way function of a random value r supplied by entity *A* and entity *B*'s private key agreement key. As discussed in Clause 8, since entity *A* could know entity *B*'s public key prior to choosing the value r , entity *A* may select approximately s bits of the established key, at the cost of generating 2^s candidate values for r in the interval between discovering entity *B*'s public key and sending KT_{A1} .
- 5) The (optional) *TVP* prevents replay of the key token from entity *A* to entity *B*.
- 6) Example: examples of this key agreement mechanism (known as Nyberg-Rueppel key agreement) are described in Annex C.4 and D.4
- 7) Public key certificates: if *Text1* is used to transfer entity *A*'s public key certificate, then requirement 2 at the beginning of this clause can be relaxed to the requirement that entity *B* is in possession of an authenticated copy of the CA's public verification key.

10.4 Key agreement mechanism 4

This key agreement mechanism establishes in two passes a shared secret key between entities *A* and *B* with joint key control without prior exchange of keying information. This mechanism provides neither entity authentication nor key authentication.

See Figure 4 for a diagram of key agreement mechanism 4.

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in H , computes $F(r_A, g)$, constructs the key token

$$KT_{A1} = F(r_A, g) \parallel \text{Text1}$$

and sends it to entity *B*.

Key Token Construction (B1) Entity *B* randomly and secretly generates r_B in H , computes $F(r_B, g)$, constructs the key token

$$KT_{B1} = F(r_B, g) \parallel \text{Text2}$$

and sends it to entity *A*.

Key Construction (A2) Entity *A* extracts $F(r_B, g)$ from the received key token KT_{B1} and computes the shared secret key

$$K_{AB} = F(r_A, F(r_B, g)).$$

Key Construction (B2) Entity *B* extracts $F(r_A, g)$ from the received key token KT_{A1} and computes the shared secret key

$$K_{AB} = F(r_B, F(r_A, g)).$$

NOTES - This Key Agreement Mechanism has the following properties:

- 1) Number of passes: 2.
- 2) Key authentication: this mechanism does not provide key authentication. However, this mechanism may be useful in environments where the authenticity of the key tokens is verified using other means. For instance, a hash-code of the key tokens may be exchanged between the entities using a second communication channel. See also Public Key Transport Mechanism 2.
- 3) A separate channel or means exists whereby the key tokens can be verified.
- 4) Key confirmation: this mechanism provides no key confirmation.
- 5) This is a key agreement mechanism, since the established key is a one-way function of random values r_A and r_B supplied by entities *A* and *B* respectively. As discussed in Clause 8, since entity *B* could know $F(r_A, g)$ prior to choosing the value r_B , entity *B* may select approximately s bits of the established key, at the cost of generating 2^s candidate values for r_B in the interval between receiving KT_{A1} and sending KT_{B1} .
- 6) Example: examples of this mechanism (known as Diffie-Hellman key agreement) are described in Annexes C.5 and D.6.

10.5 Key agreement mechanism 5

This key agreement mechanism establishes in two passes a shared secret key between entities *A* and *B* with mutual implicit key authentication and joint key control. The following requirements shall be satisfied:

- 1) Each entity *X* has a private key agreement key h_X in H and a public key agreement key $p_X = F(h_X, g)$.
- 2) Each entity has access to an authenticated copy of the public key agreement key of the other entity. This may be achieved using the mechanisms of Clause 12.
- 3) Both entities have agreed on a common one-way function w .

See Figure 4 for a diagram of key agreement mechanism 5.

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in H , computes $F(r_A, g)$ and sends the key token

$$KT_{A1} = F(r_A, g) \parallel \text{Text1}$$

to entity *B*.

Key Token Construction (B1) Entity *B* randomly and secretly generates r_B in H , computes $F(r_B, g)$ and sends the key token

$$KT_{B1} = F(r_B, g) \parallel \text{Text2}$$

to entity *A*.

Key Construction (B2) Entity *B* extracts $F(r_A, g)$ from the received key token KT_{A1} and computes the shared secret key as

$$K_{AB} = w(F(h_B, F(r_A, g)) \parallel F(r_B, p_A))$$

where w is a one-way function.

Key Construction (A2) Entity *A* extracts $F(r_B, g)$ from the received key token KT_{B1} and computes the shared secret key as

$$K_{AB} = w(F(r_A, p_B) \parallel F(h_A, F(r_B, g)))$$

NOTES - This Key Agreement Mechanism has the following properties:

- 1) Number of passes: 2.
- 2) Key authentication: this mechanism provides mutual implicit key authentication. If the data field *Text2* contains a *MAC* (on known data) computed using the key K_{AB} , then this mechanism provides explicit key authentication from entity *B* to entity *A*.
- 3) Key confirmation: if the data field *Text2* contains a *MAC* (on known data) computed using the key K_{AB} , then this mechanism provides key confirmation from entity *B* to entity *A*.
- 4) This is a key agreement mechanism since the established key is a one-way function of random values r_A and r_B supplied by entities *A* and *B* respectively.
- 5) Example: examples of this key agreement mechanism (known as the Matsumoto-Takahshima-Imai $A(0)$ key agreement scheme) are described in Annexes C.6 and D.5. Another example is known as the Goss protocol.
- 6) Public key certificates: if *Text1* and *Text2* contain the public key certificates of entity *A*'s and *B*'s key agreement key, respectively, then the requirement 2 at the beginning of this clause can be replaced by the requirement that each entity is in possession of an authenticated copy of the *CA*'s public verification key.
- 7) It is possible under certain circumstances that this mechanism is subject to a source substitution attack [18]. If this is a concern, this type of attack can be avoided by ensuring that as part of the process of submitting a public key to a *CA* for certification, the submitter proves possession of the corresponding private key. See Clauses 7.2.2 and 8.3.3 of ISO/IEC 15945:2002 for a process on proof of possession. This type of attack is slightly more serious against elliptic curve protocols [14].

10.6 Key agreement mechanism 6

This key agreement mechanism establishes in two passes a shared secret key between entities *A* and *B* with mutual implicit key authentication and joint key control. It is based on the use of both an asymmetric encipherment and signature system. The following requirements shall be satisfied:

- 1) Entity *A* has an asymmetric encipherment system with the transformations (E_A, D_A) .
- 2) Entity *B* has an asymmetric signature system with the transformations (S_B, V_B) .
- 3) Entity *A* has access to an authenticated copy of entity *B*'s public verification transformation V_B . This may be achieved using the mechanisms of Clause 12.
- 4) Entity *B* has access to an authenticated copy of entity *A*'s public encipherment transformation E_A . This may be achieved using the mechanisms of Clause 12.

See Figure 5 for a diagram of key agreement mechanism 6.

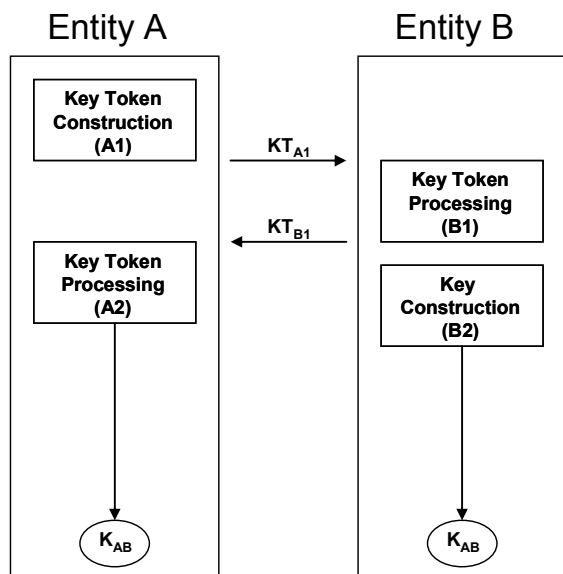


Figure 5: Key Agreement Mechanism 6

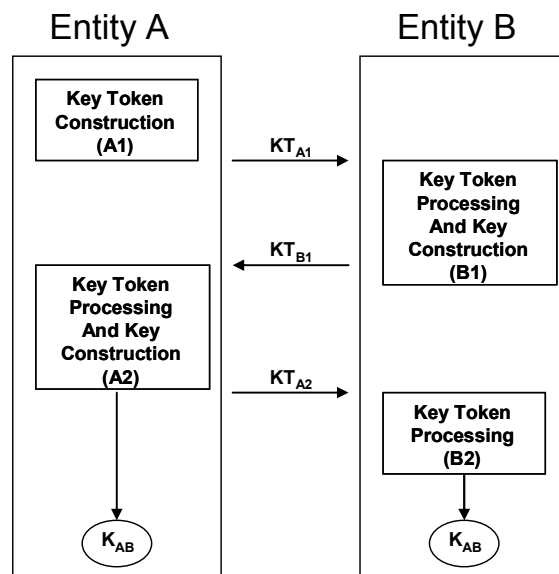


Figure 6: Key Agreement Mechanism 7

Key Token Construction (A1) Entity A generates a random number r_A and sends the key token

$$KT_{A1} = r_A \parallel \text{Text1}$$

to entity B.

Key Token Processing (B1) Entity B generates a random number r_B and signs a data block consisting of the distinguishing identifier for entity A, the random number r_A , the random number r_B and some optional data *Text2* using its private signature transformation S_B

$$BS = S_B(A \parallel r_A \parallel r_B \parallel \text{Text2}).$$

Then entity B enciphers a data block consisting of its distinguishing identifier for entity B (optional), the signed block *BS*, and some optional data *Text3* using entity A's public encipherment transformation E_A . Entity B then sends the key token

$$KT_{B1} = E_A(B \parallel BS \parallel \text{Text3}) \parallel \text{Text4}$$

back to entity A.

Key Construction (B2) The shared secret key consists of all or part of entity B's signature Σ contained in the signed block *BS* (see Note 1 in Clause 4), used with a key derivation function.

Key Token Processing (A2) Entity A decipheres the key token KT_{B1} using its private decipherment transformation D_A , optionally checks the sender identifier for entity B, and uses entity B's public verification transformation V_B to verify the digital signature of the signed block *BS*. Then entity A checks the recipient identifier for entity A and consistency of the random number r_A in the signed block *BS* with the random number r_A sent in token KT_{A1} . If all checks are successful, entity A accepts all or part of entity B's signature of the signed block *BS* used with a key derivation function as the shared secret key.

NOTES - This Key Agreement Mechanism has the following properties:

- 1) Number of passes: 2.
- 2) In the key construction processing, the part of the signature Σ to be used as the shared secret key between entities *A* and *B* must be previously established or be dependant on the signature scheme used.
- 3) Key authentication: this mechanism provides implicit key authentication from entity *A* to entity *B* and explicit key authentication from entity *B* to entity *A*.
- 4) Key confirmation: If the data field *Text3* contains a MAC (on known data) computed using the key K_{AB} , then this mechanism provides key confirmation from entity *B* to entity *A*.
- 5) This is a key agreement mechanism, since the established key is a one-way function of random values r_A and r_B supplied by entities *A* and *B* respectively. As discussed in Clause 8, since entity *B* could know $F(r_A, g)$ prior to choosing the value r_B , entity *B* may select approximately s bits of the established key, at the cost of generating 2^s candidate values for r_B in the interval between receiving KT_{A1} and sending KT_{B1} .
- 6) Example: this mechanism is derived from Beller and Yacobi's two pass protocol described in Annex C.7.
- 7) Public key certificates: if *Text1* and *Text4* contain the public key certificate of entity *A*'s encipherment key and the public key certificate of entity *B*'s verification key, respectively, then the requirements 3 and 4 at the beginning of this clause can be relaxed to the requirement that each entity is in possession of an authenticated copy of the CA's public verification key.
- 8) A significant feature of this scheme is that the identity of entity *B* may remain anonymous to eavesdroppers, of particular advantage in the wireless environment which is a main environment for the application of this scheme.

10.7 Key agreement mechanism 7

This key agreement mechanism is based on the three-pass authentication mechanism of ISO/IEC 9798-3 and establishes in three passes a shared secret key between entities *A* and *B* with mutual authentication. The following requirements shall be satisfied:

- 1) Each entity *X* has an asymmetric signature system (S_X, V_X) .
- 2) Each entity has access to an authenticated copy of the public verification transformation of the other entity. This may be achieved using the mechanisms of Clause 12.
- 3) Each entity has a common MAC function.

See Figure 6 for a diagram of key agreement mechanism 7.

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in H , computes $F(r_A, g)$, constructs the key token

$$KT_{A1} = F(r_A, g) || \text{Text1}$$

and sends it to entity *B*.

Key Token Processing and Key Construction (B1) Entity *B* randomly and secretly generates r_B in H , computes $F(r_B, g)$, computes the shared secret key as

$$K_{AB} = F(r_B, F(r_A, g)),$$

constructs the signed key token

$$KT_{B1} = S_B(DB_1) || MAC_{K_{AB}}(DB_1) || \text{Text3}$$

where

$$DB_1 = F(r_B, g) || F(r_A, g) || A || \text{Text2}$$

and sends it back to entity A.

Key confirmation is provided by sending $MAC_{K_{AB}}(DB_1)$ in KT_{B1} . Alternatively, if both parties have a common symmetric encryption system, key confirmation can be obtained by encrypting part of the token by replacing KT_{B1} with

$$KT_{B1} = F(r_B, g) || E_{K_{AB}}(S_B(DB_1)).$$

Key Token Processing (A2) Entity A verifies entity B's signature on the key token KT_{B1} using entity B's public verification key, verifies entity A's distinguishing identifier and the value $F(r_A, g)$ sent in step (A1). If the check is successful, entity A proceeds to compute the shared secret key as

$$K_{AB} = F(r_A, F(r_B, g)).$$

Using K_{AB} , entity A verifies $MAC_{K_{AB}}(DB_1)$. Then entity A constructs the signed key token

$$KT_{A2} = S_A(DB_2) || MAC_{K_{AB}}(DB_2) || \text{Text5}$$

where

$$DB_2 = F(r_A, g) || F(r_B, g) || B || \text{Text4}$$

and sends it to entity B.

Key confirmation is provided by sending $MAC_{K_{AB}}(DB_2)$ in KT_{A2} . Alternatively, key confirmation can be obtained by encrypting part of the token by replacing KT_{A2} with

$$KT_{A2} = E_{K_{AB}}(S_A(DB_2)).$$

Key Token Processing (B2) Entity B verifies entity A's signature on the key token KT_{A2} , using entity A's public verification key, then verifies entity B's distinguishing identifier and that the values $F(r_A, g)$ and $F(r_B, g)$ agree with the values exchanged in the previous steps. If the check is successful, entity B verifies $MAC_{K_{AB}}(DB_2)$ using

$$K_{AB} = F(r_B, F(r_A, g)).$$

NOTES - This Key Agreement Mechanism has the following properties:

- 1) Number of passes: 3.
- 2) Key and entity authentication: this mechanism provides mutual explicit key authentication and mutual entity authentication.
- 3) Key confirmation: this mechanism provides mutual key confirmation.