



BSI Standards Publication

Information technology — Security techniques — Key management

Part 3: Mechanisms using asymmetric techniques

National foreword

This British Standard is the UK implementation of ISO/IEC 11770-3:2015+A1:2017. It supersedes BS ISO/IEC 11770-3:2015, which is withdrawn.

The start and finish of text introduced or altered by amendment is indicated in the text by tags. Tags indicating changes to ISO/IEC text carry the number of the ISO/IEC amendment. For example, text altered by ISO/IEC amendment 1 is indicated by A1 A1.

The UK participation in its preparation was entrusted to Technical Committee IST/33/2, Cryptography and Security Mechanisms.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2018
Published by BSI Standards Limited 2018

ISBN 978 0 580 96554 8

ICS 35.030

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2018.

Amendments/corrigenda issued since publication

Date	Text affected
31 January 2018	Implementation of ISO/IEC amendment A1:2017

INTERNATIONAL
STANDARD

ISO/IEC
11770-3

Third edition
2015-08-01

**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 11770-3:2015(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2015



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2015, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

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

Contents

Page

Foreword	v
Introduction	vi
1 Scope	1
2 Normative references	1
3 Terms and definitions	2
4 Symbols and abbreviations	6
5 Requirements	9
6 Key derivation functions	9
7 Cofactor multiplication	9
8 Key commitment	10
9 Key confirmation	10
10 Framework for key management	12
10.1 General	12
10.2 Key agreement between two parties	12
10.3 Key agreement between three parties	12
10.4 Secret key transport	13
10.5 Public key transport	13
11 Key agreement	13
11.1 Key agreement mechanism 1	13
11.2 Key agreement mechanism 2	15
11.3 Key agreement mechanism 3	15
11.4 Key agreement mechanism 4	17
11.5 Key agreement mechanism 5	18
11.6 Key agreement mechanism 6	19
11.7 Key agreement mechanism 7	20
11.8 Key agreement mechanism 8	22
11.9 Key agreement mechanism 9	22
11.10 Key agreement mechanism 10	23
11.11 Key agreement mechanism 11	24
11.12 Key agreement mechanism 12	25
11.13 Key agreement mechanism 13	26
11.14 Key agreement mechanism 14	27
12 Secret key transport	28
12.1 Secret key transport mechanism 1	28
12.2 Secret key transport mechanism 2	30
12.3 Secret key transport mechanism 3	31
12.4 Secret key transport mechanism 4	33
12.5 Secret key transport mechanism 5	34
12.6 Secret key transport mechanism 6	37
13 Public key transport	38
13.1 Public key transport mechanism 1	38
13.2 Public key transport mechanism 2	39
13.3 Public key transport mechanism 3	40
Annex A (normative) Object identifiers	42
Annex B (informative) Properties of key establishment mechanisms	49
Annex C (informative) Examples of key derivation functions	52
Annex D (informative) Examples of key establishment mechanisms	59

Annex E (informative) **Examples of elliptic curve based key establishment mechanisms**..... 63

Annex F (informative) **Example of bilinear pairing based key establishment mechanisms**..... 73

Annex G (informative) **Secret key transport**..... 76

Annex H (informative) **Patent information**..... 81

Bibliography 85

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: [Foreword - Supplementary information](#)

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 27, *Security techniques*.

This third edition cancels and replaces the second edition (ISO/IEC 11770-3:2008 with ISO/IEC 11770-3/Cor1:2009), which has been technically revised.

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*
- *Part 5: Group key management*
- *Part 6: Key derivation*

Further parts may follow.

Introduction

This part of ISO/IEC 11770 describes 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 such cryptosystems is based on the presumed intractability of solving the discrete logarithm problem over certain finite fields. Other public key cryptosystems such as RSA are based on the difficulty of the integer factorization problem.

A third class of public key cryptosystems 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. When based on a carefully chosen 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 of comparable size. All known general purpose algorithms for determining elliptic curve discrete logarithms take exponential time. 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, as well as system parameters, and allows for computations using smaller integers.

This part of ISO/IEC 11770 includes mechanisms based on the following:

- finite fields;
- elliptic curves;
- bilinear pairings.

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 those in [Annex H](#).

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

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

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.

- a) Establish a shared secret key for use in a symmetric cryptographic technique between two entities *A* and *B* by key agreement. In a secret key agreement mechanism, the secret key is computed as the result of a data exchange between the two entities *A* and *B*. Neither of them should be able to predetermine the value of the shared secret key.
- b) Establish a shared secret key for use in a symmetric cryptographic technique between two entities *A* and *B* via 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.
- c) Make an entity's public key available to other entities via key transport. In a public key transport mechanism, the public key of entity *A* shall 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.^[6]

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

- key lifecycle management,
- mechanisms to generate or validate asymmetric key pairs, and
- 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 such as smart cards, etc.

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

NOTE To provide origin authentication for 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, in whole or in part, are normatively referenced in this document and are indispensable for its application. 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 11770-1, *Information technology — Security techniques — Key management — Part 1: Framework*

^{A1} ISO/IEC 11770-6, *Information technology — Security techniques — Key management — Part 6: Key derivation* ^{A1}

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

ISO/IEC 18031, *Information technology — Security techniques — Random bit generation*

^{A1} ISO/IEC 19772, *Information technology — Security techniques — Authenticated encryption* ^{A1}

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, then it is computationally infeasible to derive the private transformation

Note 1 to entry: A system based on asymmetric cryptographic techniques can either be an encryption system, a signature system, a combined encryption and signature system, or a key agreement scheme. With asymmetric cryptographic techniques there are four elementary transformations: *signature* and *verification* for signature systems, *encryption* and *decryption* for encryption systems. The signature and the decryption transformations are kept private by the owning entity, whereas the corresponding verification and encryption 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 encryption system

system based on asymmetric cryptographic techniques whose public transformation is used for encryption and whose private transformation is used for decryption

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

collision-resistant hash-function

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

[SOURCE: ISO/IEC 10118-1:2000, 3.2]

3.6

decryption

reversal of a corresponding encryption

[SOURCE: ISO/IEC 11770-1:2010, 2.6]

3.7

digital signature

data unit appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to verify 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

3.8

distinguishing identifier

information which unambiguously distinguishes an entity

[SOURCE: ISO/IEC 11770-1:2010, 2.9]

3.9

encryption

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

[SOURCE: ISO/IEC 11770-1:2010, 2.10]

3.10

entity authentication

corroboration that an entity is the one claimed

[SOURCE: ISO/IEC 9798-1:2010, 3.14]

3.11

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

assurance of the identity of entity *A* for entity *B*

3.12

explicit key authentication from entity *A* to entity *B*

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

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

3.13

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

3.14

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 1 to entry: 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.

3.15

hash-function

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

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

Note 1 to entry: Computational feasibility depends on the specific security requirements and environment.

Note 2 to entry: For the purposes of this standard all hash-functions are assumed to be collision-resistant (see 3.5).

[SOURCE: ISO/IEC 10118-1:2000, 3.5]

3.16

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

3.17

key

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

[SOURCE: ISO/IEC 11770-1:2010, 2.12]

3.18

key agreement

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

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

3.19

key commitment

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

3.20

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

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

3.21

key control

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

3.22

key derivation function

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

3.23

key establishment

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

3.24

key token

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

3.25

key transport

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

3.26

message authentication code

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

Note 1 to entry: A MAC is sometimes called a cryptographic check value (see for example ISO 7498-2[1]).

[SOURCE: ISO/IEC 9797-1:2011, 3.9]

3.27

Message Authentication Code algorithm 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:

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

Note 1 to entry: A MAC algorithm is sometimes called a cryptographic check function (see for example ISO 7498-2[1]).

Note 2 to entry: Computational feasibility depends on the user's specific security requirements and environment.

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

3.28

mutual entity authentication

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

3.29

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

3.30

one-way function

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

3.31

prefix free representation

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

3.32

private key

key of an entity's asymmetric key pair that is kept private

Note 1 to entry: The security of an asymmetric system depends on the privacy of this key.

[SOURCE: ISO/IEC 11770-1:2010, 2.35]

3.33

public key

key of an entity's asymmetric key pair which can usually be made public without compromising security

Note 1 to entry: In the case of an asymmetric signature system, the public key defines the verification transformation. In the case of an asymmetric encryption system, the public key defines the encryption transformation, conditional on the inclusion of randomisation elements. A key that is "publicly known" is not necessarily globally available. The key can only be available to all members of a pre-specified group.

[SOURCE: ISO/IEC 11770-1:2010, 2.36]

3.34

public key certificate

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

3.35

public key information

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

3.36

secret key

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

3.37

sequence number

time variant parameter whose value is taken from a specified sequence which is non-repeating within a certain time period

[SOURCE: ISO/IEC 11770-1:2010, 2.44]

3.38

signature system

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

3.39

third party forward secrecy

property that knowledge of a third party's private key subsequent to a key agreement operation does not enable an opponent to recompute previously derived keys

Note 1 to entry: Instead of third party forward secrecy, master key forward secrecy is also used in Reference [19].

3.40

time stamp

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

3.41

time-stamping authority

trusted third party trusted to provide a time-stamping service

[SOURCE: ISO/IEC 13888-1:2009, 3.58]

3.42

time variant parameter

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

Note 1 to entry: If a random number is used, then this is as a challenge in a challenge-response protocol. See also ISO/IEC 9798-1:2010, Annex B.

[SOURCE: ISO/IEC 9798-1:2010, 3.36]

3.43

trusted third party

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

[SOURCE: ISO/IEC 9798-1:2010, 3.38]

4 Symbols and abbreviations

The following symbols and abbreviations are used in this part of ISO/IEC 11770.

A, B, C distinguishing identifiers of entities

BE	encrypted data block
BS	signed data block
CA	certification authority
$Cert_A$	entity A 's public key certificate
D_A	entity A 's private decryption transformation function
d_A	entity A 's private decryption key
E	elliptic curve, either given by an equation of the form $Y^2 = X^3 + aX + b$ over the field $GF(p^m)$ for $p > 3$ and a positive integer m , by an equation of the form $Y^2 + XY = X^3 + aX^2 + b$ over the field $GF(2^m)$, or by an equation of the form $Y^2 = X^3 + aX^2 + b$ over the field $GF(3^m)$, together with an extra point O_E referred to as the point at infinity, which is denoted by $E/GF(p^m)$, $E/GF(2^m)$, or $E/GF(3^m)$, respectively
E_A	entity A 's public encryption transformation function
e_A	entity A 's public encryption key
F	key agreement function
$F(h,g)$	key agreement function using as input a factor h and a common element g
FP	key agreement function based on pairing
G	point on E with order n
g	common element shared publicly by all the entities that use the key agreement function F
$\gcd(a,b)$	greatest common divisor of two integers a and b
$GF(p^m), GF(2^m), GF(3^m)$	finite field with $p^m, 2^m, 3^m$ elements for a prime $p > 3$ and a positive integer m
h_A	entity A 's private key agreement key
hash	hash-function
j	cofactor used in performing cofactor multiplication
K	secret key for a symmetric cryptosystem
K_{AB}	secret key shared between entities A and B
NOTE 1 In practical implementations the shared secret key should be subject to further processing before it can be used for a symmetric cryptosystem.	
kdf	key derivation function
KT	key token
KT_A	entity A 's key token
KT_{Ai}	key token sent by entity A after processing phase i
l	supplementary value used in performing cofactor multiplication

M	data message
MAC	Message Authentication Code
$MAC_K(Z)$	output of a MAC algorithm when using as input the secret key K and an arbitrary data string Z
MQV	Menezes-Qu-Vanstone
n	prime divisor of the order (or cardinality) of an elliptic curve E over a finite field
O_E	elliptic curve point at infinity
P	point on an elliptic curve E
p_A	entity A 's public key-agreement key
pairing	pairing defined over an elliptic curve and used in FP
<i>parameters</i>	parameters used in the key derivation function
PKI_A	entity A 's public key information
P_X	public key-agreement key in an elliptic curve of entity X
q	prime power p^m for some prime $p \neq 3$ and some integer $m \geq 1$
r	random number generated in the course of a mechanism
r_A	random number issued by entity A in a key agreement mechanism
S_1, S_2, S_3	sets of elements
S_A	entity A 's private signature transformation function
s_A	entity A 's private signature key
T	trusted third party
$Text_i$	i th optional text, data or other information that may be included in a data block, if desired
TVP	time-variant parameter such as a random number, a time stamp, or a sequence number
V_A	entity A 's public verification transformation function
v_A	entity A 's public verification key
w	one-way function
$X(P)$	x-coordinate of a point P
$\sqrt{q}$	square root of a positive number q
$\#E$	order (or cardinality) of an elliptic curve E
$\parallel$	concatenation of two data elements
$\lceil x \rceil$	smallest integer greater than or equal to the real number x
Σ	digital signature

$\pi(P)$ $(X(P) \bmod 2^{\lceil \rho/2 \rceil}) + 2^{\lceil \rho/2 \rceil}$ where $\rho = \lceil \log_2 n \rceil$ and $X(P)$ is the x -coordinate of the point P

NOTE 2 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 3 The keys of an asymmetric cryptosystem are denoted by lower case letters (indicating its function) 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 involved in a mechanism are aware of each other's claimed identities. This may be achieved by the inclusion of identifiers in information exchanged between the two entities, or it may be apparent from the context of use of the mechanism. Verifying the identity means checking that a received identifier field agrees with some known (trusted) or expected value.

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 will often be the case that the form of a shared secret established as a result of using a mechanism specified in this part of ISO/IEC 11770 will not conform to the form needed for a specific cryptographic algorithm, so some processing will be needed. Moreover, 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 therefore advisable to pass the shared secret through a key derivation function, e.g. involving the use of a hash function. The use of an inadequate key derivation function could compromise the security of the key agreement scheme with which it is used. It is recommended to use a one-way function as a key derivation function.

A key derivation function produces keys that are computationally indistinguishable from randomly generated keys. The key derivation function takes as input a shared secret and a set of key derivation parameters and produces 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 shall be agreed upon (see ISO/IEC 11770-6 for further guidance on key derivation functions).

[Annex C](#) provides examples of key derivation functions.

7 Cofactor multiplication

This clause applies only to mechanisms using elliptic curve cryptography. The key agreement mechanisms in [Clause 11](#) and the key transport mechanisms in [Clauses 12](#) and [13](#) 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. One example of such an attack is known as the 'small subgroup attack'.

NOTE 1 The small subgroup attack is described in [\[37\]](#).

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, as specified in ISO/IEC 11770-1.

As an alternative to public key validation, a technique called cofactor multiplication as specified in [Clause 11](#) can be used. The values j and l , defined below, are used in cofactor multiplication.

If cofactor multiplication is used, there are two options:

- If compatibility with entities not using cofactor multiplication is not required, then let $j = \#E / n$ and $l = 1$. If this option is chosen, both parties involved shall agree to use this option; otherwise the mechanism will not work.
- If compatibility with entities not using cofactor multiplication is required, then let $j = \#E / n$ and $l = j^{-1} \bmod n$.

NOTE 2 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 required, then let $j = l = 1$.

Regardless of whether or not cofactor multiplication is used, 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 procedure has failed.

It is particularly appropriate to perform public key validation or cofactor multiplication in the following cases:

- if the entity’s public key is not authenticated;
- if the key token is not authenticated;
- if the user’s public key is intended for a long-term use.

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 necessary to perform these tests.

8 Key commitment

[Clause 11](#) describes key agreement mechanisms in which the established key is the result of applying a one-way function to 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. As a result, such an 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.^[31]

One 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. Key commitment can be performed by having the first entity hash the public key or key token and send the hash-code to the second entity; the second entity then replies with its public key or key token, and the first entity replies with its public key or key token. The second entity can now hash it and verify that the result is equal to the hash-code sent earlier.

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 (with very high probability) 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.

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 the entity performing 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" security properties during the key establishment process and may be warranted if a conservative protocol design is deemed appropriate.

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 11](#) and [12](#) 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.

- Entity *B* forms the message *M*, an 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 (omitted if not present), the octet string KT_A corresponding to entity *A*'s key token (omitted if not present), the octet string p_B corresponding to entity *B*'s public key-establishment key (omitted if not present), the octet string p_A corresponding to entity *A*'s public key-establishment key (omitted if not present) and, if present, optional additional Text1, i.e.:

$M = 02||B||A||KT_B||KT_A||p_B||p_A||\text{Text1}$, where 0x02 is the message number.

- Entity *B* calculates $K_B = \text{kdf}(K_{AB})$, and then calculates $\text{MAC}_{K_B}(M)$ for the message *M* under the (supposedly) shared secret key K_B for an appropriate MAC scheme.
- Entity *B* sends the message *M* and $\text{MAC}_{K_B}(M)$ to entity *A*.
- Entity *A* calculates $K_A = \text{kdf}(K_{AB})$, computes $\text{MAC}_{K_A}(M)$ using the received message *M*, and verifies $\text{MAC}_{K_B}(M) = \text{MAC}_{K_A}(M)$.
- 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'* as 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 (omitted if not present), the octet string KT_B corresponding to entity *B*'s key token (omitted if not present), the octet string p_A corresponding to entity *A*'s public key-establishment key (omitted if not present), the octet string p_B corresponding to entity *B*'s public key-establishment key (omitted if not present) and optional additional octet string Text2, i.e.:

$M' = 03||A||B||KT_A||KT_B||p_A||p_B||\text{Text2}$, where 0x03 is the message number.

- Entity *A* calculates $\text{MAC}_{K_A}(M')$ under the (supposedly) shared secret K_A using an appropriate MAC scheme.
- Entity *A* sends *M'* and $\text{MAC}_{K_A}(M')$ to entity *B*.
- 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.

10 Framework for key management

10.1 General

This clause contains a high-level description of a framework for the key establishment mechanisms specified in this part of ISO/IEC 11770. Four categories of mechanism are defined (key agreement between two parties, key agreement between three parties, secret key transport and public key transport), together with requirements for their use.

10.2 Key agreement between two parties

^{A1} The provisions in this subclause apply to key agreement mechanisms 11.1 to 11.11, 11.13 and 11.14, all of which specify mechanisms for key agreement between two parties. ^{A1} Key Agreement between two parties 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.

Key agreement between two entities *A* and *B* takes place in a context shared by the two entities. The context consists of sets S_1 and S_2 , and a key agreement function *F*. The function *F* shall satisfy the following requirements:

- a) $F : S_1 \times S_2 \rightarrow S_2$ maps elements $(h, g) \in S_1 \times S_2$ to S_2 , and we write $y = F(h, g)$.
- b) *F* satisfies the commutativity condition $F(h_A, F(h_B, g)) = F(h_B, F(h_A, g))$.
- c) 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.
- d) The entities *A* and *B* share a common element g in S_2 which may be publicly known.
- e) The entities acting in this setting can efficiently compute function values $F(h, g)$ and can efficiently generate random elements in S_1 . Depending on the particular key agreement mechanism, further conditions may be imposed.

NOTE 1 Examples for the function *F* are given in Annex D and Annex E. See also ISO/IEC 15946-1.

NOTE 2 As discussed in Clause 6, in practical implementations of the key agreement mechanisms the shared secret key should be subject to further processing.

NOTE 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.

10.3 Key agreement between three parties

This clause applies to the Key Agreement mechanism 11.12 that describes key agreement between three parties. Key agreement between three parties is the process of establishing a shared secret key among three entities *A*, *B*, and *C* in such a way that none of them can predetermine the value of the shared secret key. Key agreement among three entities *A*, *B*, and *C* takes place in a context shared by the three entities. The context consists of sets S_1 , S_2 , and S_3 , a function *F*, and a function *FP*. The functions *F* and *FP* shall satisfy the following requirements:

- $F : S_1 \times S_2 \rightarrow S_2$ maps elements $(h, g) \in S_1 \times S_2$ to S_2 , and we write $y = F(h, g)$.
- *F* satisfies the commutativity condition $F(h_A, F(h_B, g)) = F(h_B, F(h_A, g))$.
- 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.

- $FP : S_1 \times S_2 \times S_2 \rightarrow S_3$ maps an element $(h_C, F(h_A, g), F(h_B, g))$ in $S_1 \times S_2 \times S_2$ to an element of S_3 , and we write $z = FP(h_C, F(h_A, g), F(h_B, g))$. ISO/IEC 15946-1 shall be referred for the relation between F and FP .
- FP satisfies the commutativity condition
 - $FP(h_C, F(h_A, g), F(h_B, g)) = FP(h_C, F(h_B, g), F(h_A, g)) = FP(h_B, F(h_A, g), F(h_C, g))$
 $= FP(h_A, F(h_B, g), F(h_C, g)) = FP(h_A, F(h_C, g), F(h_B, g)) = FP(h_B, F(h_C, g), F(h_A, g))$.
- It is computationally intractable to find $FP(h_C, F(h_A, g), F(h_B, g))$ from $F(h_A, g)$, $F(h_B, g)$, $F(h_C, g)$, and g . This implies that $F(\cdot, p_A, p_B)$ is a one-way function.
- The entities A , B , and C share a common element g in S_2 which may be publicly known.
- The entities acting on this setting can efficiently compute function values $F(h, g)$ and $FP(h_C, F(h_B, g), F(h_A, g))$, and can efficiently generate random elements in S_1 . Depending on the particular key agreement mechanism, further conditions may be imposed.

NOTE 4 An example of a possible function FP is given in [Annex F](#).

NOTE 5 As discussed in [Clause 6](#), 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 by: (1) by extracting bits from the shared secret key K_{ABC} directly, or (2) by passing the shared secret key K_{ABC} and optionally other nonsecret data through a one-way function and extracting bits from the output.

10.4 Secret key transport

Secret key transport (often abbreviated to "key transport") is the process of transferring a secret key, chosen by one entity (or a trusted centre), to another entity, suitably protected by asymmetric cryptographic encryption.

10.5 Public key transport

Public key transport makes an entity's public key available to other entities in an authenticated fashion. Authenticated distribution of public keys is an essential security requirement. This distribution can be achieved in two main ways:

- a) Public key distribution without a trusted third party.
- b) Public key distribution involving a trusted third party, such as a certification authority.

The public key of an entity A is part of the public key information of entity A . The public key information includes at least entity A 's distinguishing identifier and entity A 's public key.

11 Key agreement

11.1 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:

- Each entity X has a private key agreement key h_X in S_1 and a public key agreement key $p_X = F(h_X, g)$.
- 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 described in [Clause 13](#).

Key agreement mechanism 1 is summarised in [Figure 1](#).

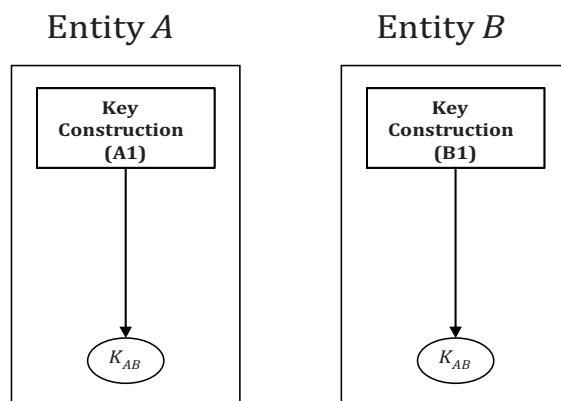


Figure 1 — Key Agreement Mechanism 1

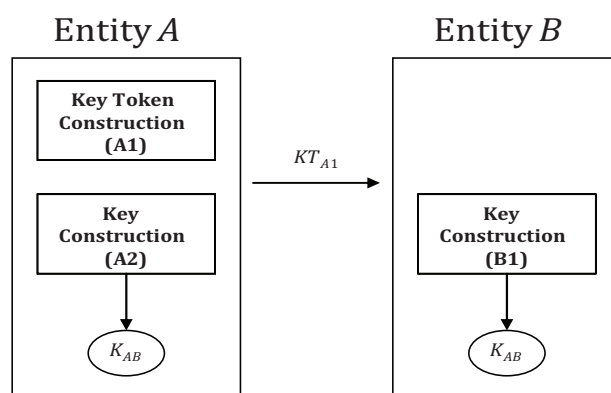


Figure 2 — Key Agreement Mechanisms 2, 8

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 requirements on F specified in [Clause 10](#), the two computed values for the key K_{AB} are identical.

NOTE 1 The number of passes is 0.

NOTE 2 This mechanism provides mutual implicit key authentication. However, a zero-pass protocol such as this will always generate the same key. One way to eliminate this problem is to ensure that the key is only used once. Furthermore, the use of a unique initialization vector with each utilization of the key can also solve this problem.

NOTE 3 This mechanism does not provide key confirmation.

NOTE 4 This mechanism 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 might learn the other entity's public key prior to choosing their private key. As described in [Clause 8](#), such an entity can 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.

NOTE 5 Examples of this mechanism (known as Diffie-Hellman key agreement) are given in Annexes [D.2](#), [D.3](#), and [E.3](#)

11.2 Key agreement mechanism 2

This key agreement mechanism establishes a shared secret key in one pass 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:

- Entity *B* has a private key agreement key h_B in S_1 and a public key agreement key $p_B = F(h_B, g)$.
- 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 described in [Clause 13](#).

Key agreement mechanism 2 is summarised in [Figure 2](#).

Key Token Construction (A1) Entity *A* randomly and secretly generates r in S_1 , computes $F(r, g)$ and sends the key token $KT_{A1} = F(r, g) || \text{Text}$ to entity *B*

Key Construction (A2) Entity *A* computes the shared 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 $K_{AB} = F(h_B, F(r, g))$.

As a consequence of the requirements on F specified in [Clause 10](#), the two computed values for the key K_{AB} are identical.

NOTE 1 The number of passes is 1.

NOTE 2 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).

NOTE 3 This mechanism does not provide key confirmation.

NOTE 4 This mechanism 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 learn 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} .

NOTE 5 Examples of this mechanism (known as ElGamal key agreement) are described in Annexes [D.4](#) and [E.4](#).

NOTE 6 As entity *B* receives the information necessary to compute the key K_{AB} from entity *A*, which has not been authenticated, use of K_{AB} by entity *B* should be restricted to functions not requiring trust in entity *A*'s authenticity, such as decryption and generation of message authentication codes.

11.3 Key agreement mechanism 3

This key agreement mechanism establishes a shared secret key in one pass 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:

- Entity *A* has an asymmetric signature system (S_A, V_A) .
- Entity *B* has access to an authenticated copy of the public verification transformation V_A . This may be achieved using the mechanisms described in [Clause 13](#).
- Entity *B* has a key agreement scheme with keys (h_B, p_B) .
- 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 described in [Clause 13](#).
- (Optional) If used, 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.

- The entities *A* and *B* have agreed on a MAC function and a way to use K_{AB} as the key for this MAC function. ISO/IEC 9797[5] is referred for a MAC function.

Key agreement mechanism 3 is summarised in Figure 3.

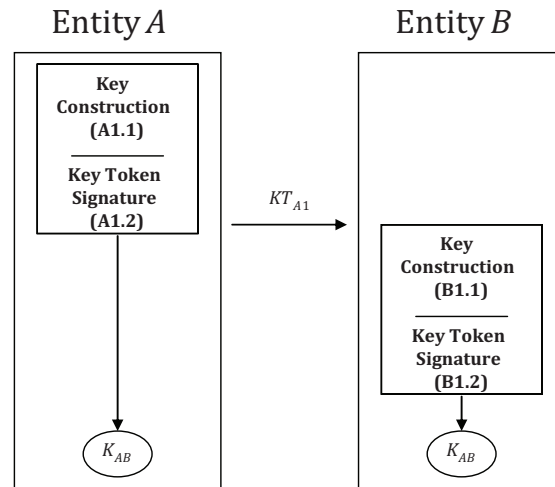


Figure 3 — Key Agreement Mechanism 3

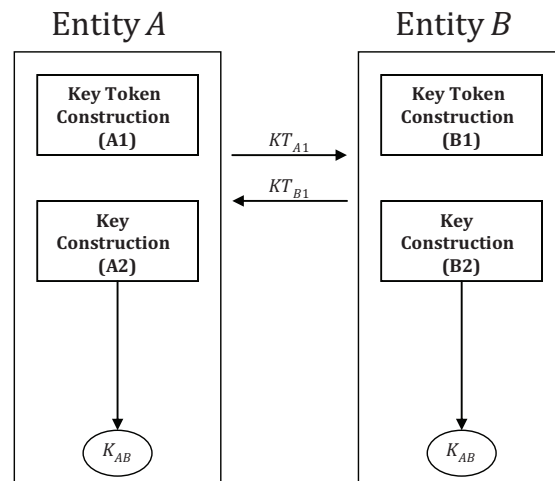


Figure 4 — Key Agreement Mechanisms 4, 5, 9

Key Construction (A1.1) Entity *A* randomly and secretly generates r in S_1 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 TVP, a time stamp or a 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, i.e.

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

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).

NOTE 1 The number of passes is 1.

NOTE 2 This mechanism provides explicit key authentication from entity A to entity B and implicit key authentication from entity B to entity A .

NOTE 3 This mechanism provides key confirmation from entity A to entity B .

NOTE 4 This mechanism 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 learn entity B 's public key prior to choosing the value r , entity A can 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} .

NOTE 5 The (optional) TVP prevents replay of the key token from entity A to entity B .

NOTE 6 Examples of this mechanism (known as Nyberg-Rueppel key agreement) are described in Annex [D.5](#) and [E.5](#).

NOTE 7 If Text1 is used to transfer entity A 's public key certificate, then requirement 2 at the beginning of [11.3](#) can be relaxed to the requirement that entity B is in possession of an authenticated copy of the CA's public verification key.

11.4 Key agreement mechanism 4

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

Key agreement mechanism 4 is summarised in [Figure 4](#).

Key Token Construction (A1) Entity A randomly and secretly generates r_A in S_1 , 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 Construction (B1) Entity B randomly and secretly generates r_B in S_1 , computes $F(r_B, g)$, constructs the key token $KT_{B1} = F(r_B, g) || \text{Text2}$, 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 $K_{AB} = F(r_B, F(r_A, g))$.

NOTE 1 The number of passes is 2.

NOTE 2 This mechanism does not provide implicit or explicit key authentication. However, this mechanism can be useful in environments where authenticity of the key tokens is verified using other means. For instance, a hash-code of the key tokens could be exchanged between the entities using a second communication channel. See also Public Key Transport Mechanism 2. Another example of entity authentication is using mechanisms specified in [\[6\]](#).

NOTE 3 A separate channel or means shall exist whereby the key tokens can be verified.

NOTE 4 This mechanism provides no key confirmation.

NOTE 5 This mechanism 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 learn $F(r_A, g)$ prior to choosing the value r_B , entity B can 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} .

NOTE 6 Examples of this mechanism (known as Diffie-Hellman key agreement) are described in Annexes [D.6](#) and [E.7](#).

11.5 Key agreement mechanism 5

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

- Each entity X has a private key agreement key h_X in S_1 and a public key agreement key $p_X = F(h_X, g)$.
- 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 described in [Clause 13](#).
- Both entities have agreed on a common one-way function w .

Key agreement mechanism 5 is summarised in [Figure 4](#).

Key Token Construction (A1) Entity A randomly and secretly generates r_A in S_1 , computes $F(r_A, g)$ and sends the key token $KT_{A1} = F(r_A, g) || \text{Text1}$ to entity B .

Key Token Construction (B1) Entity B randomly and secretly generates r_B in S_1 , computes $F(r_B, g)$ and sends the key token $KT_{B1} = F(r_B, g) || \text{Text2}$ 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 $K_{AB} = w(F(h_B, F(r_A, g)) || F(r_B, p_A))$ 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 $K_{AB} = w(F(r_A, p_B) || F(h_A, F(r_B, g)))$.

NOTE 1 The number of passes is 2.

NOTE 2 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 .

NOTE 3 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 .

NOTE 4 This mechanism 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.

NOTE 5 Examples of this key agreement mechanism (known as the Matsumoto-Takashima-Imai A(0) key agreement scheme) are described in Annexes [D.7](#) and [E.6](#). Another example is known as the Goss protocol.

NOTE 6 If Text1 and Text2 contain the public key certificates of entity A 's and B 's key agreement keys, respectively, then the requirement 2 at the beginning of [11.5](#) can be replaced by the requirement that each entity is in possession of an authenticated copy of the CA's public verification key.

NOTE 7 Under certain circumstances this mechanism may be subject to a source substitution attack.^[30] 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. This type of attack is slightly more serious in the case of protocols based on elliptic curves.^[26]

11.6 Key agreement mechanism 6

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

- Entity *A* has an asymmetric encryption system with transformations (E_A , D_A).
- Entity *B* has an asymmetric signature system with transformations (S_B , V_B).
- Entity *A* has access to an authenticated copy of entity *B*'s public verification transformation V_B . This may be achieved using the mechanisms described in [Clause 13](#).
- Entity *B* has access to an authenticated copy of entity *A*'s public encryption transformation E_A . This may be achieved using the mechanisms described in [Clause 13](#).

Key agreement mechanism 6 is summarised in [Figure 5](#).

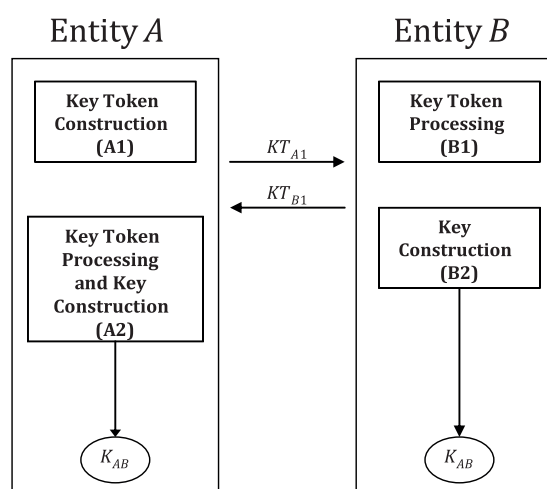


Figure 5 — Key Agreement Mechanism 6

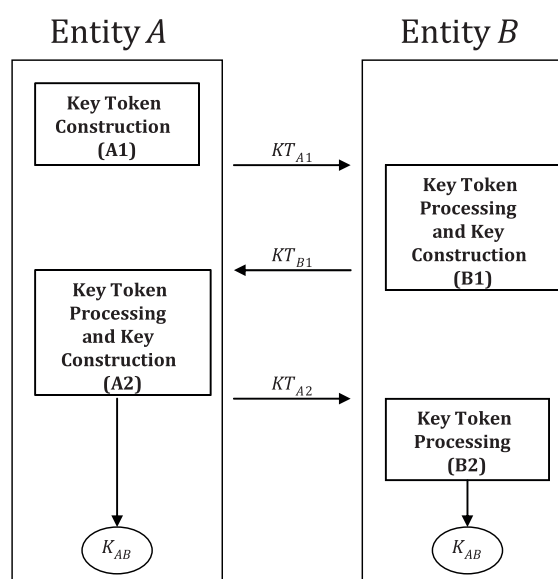


Figure 6 — Key Agreement Mechanism 7

Key Token Construction (A1) Entity *A* generates a random number r_A , constructs the key token $KT_{A1} = r_A || \text{Text1}$, and sends it 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 , to obtain $BS = S_B(A||r_A||r_B||\text{Text2})$.

Entity *B* then encrypts a data block consisting of its distinguishing identifier (optional), the signed block BS , and some optional data Text3 using entity *A*'s public encryption transformation E_A . Entity *B* then sends the key token $KT_{B1} = E_A(BS||\text{Text3})||\text{Text4}$ back to entity *A*, or entity *B* may include the identifier for *B* as $KT_{B1} = E_A(B||BS||\text{Text3})||\text{Text4}$.

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 2 in [Clause 4](#)), after passing through a key derivation function.

Key Token Processing and Key Construction (A2) Entity *A* decrypts the key token KT_{B1} using its private decryption transformation D_A , optionally checks the sender identifier, 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 and that the random number r_A in the signed block BS equals 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.

NOTE 1 The number of passes is 2.

NOTE 2 The part of the signature Σ that is to be used as the basis of the secret key established between entities *A* and *B* shall be agreed in advance.

NOTE 3 This mechanism provides implicit key authentication from entity *A* to entity *B* and explicit key authentication from entity *B* to entity *A*.

NOTE 4 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*.

NOTE 5 This mechanism 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 learn $F(r_A, g)$ prior to choosing the value r_B , entity *B* can 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} .

NOTE 6 This mechanism is derived from Beller and Yacobi's two pass protocol described in Annex [D.8](#).

NOTE 7 If Text1 and Text4 contain a public key certificate for entity *A*'s encryption key and a public key certificate for entity *B*'s verification key, respectively, then the requirements 3 and 4 at the beginning of [11.6](#) can be relaxed to the requirement that each entity is in possession of an authenticated copy of the CA's public verification key.

NOTE 8 A significant feature of this scheme is that the identity of entity *B* can remain anonymous to eavesdroppers, a property of potential significance in a wireless communication environment.

11.7 Key agreement mechanism 7

This key agreement mechanism is based on the three-pass authentication mechanism of ISO/IEC 9798-3[6] and establishes a shared secret key between entities *A* and *B* with mutual authentication. The following requirements shall be satisfied:

- Each entity *X* has an asymmetric signature system (S_X, V_X).
- Both entities have access to an authenticated copy of the public verification transformation of the other entity. This may be achieved using the mechanisms described in [Clause 13](#).
- The two entities have agreed on a common MAC function.

Key agreement mechanism 7 is summarised in [Figure 6](#).

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in S_1 , 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 S_1 , computes $F(r_B, g)$, computes the shared secret key as $K_{AB} = F(r_B, F(r_A, g))$, and constructs the signed key token

$$KT_{B1} = S_B(DB_1) || MAC_{KAB}(DB_1) || Text3,$$

where $DB_1 = F(r_B, g) || F(r_A, g) || A || Text2$, and sends it back to entity *A*.

Key confirmation is provided by including $MAC_{KAB}(DB_1)$ in KT_{B1} . Alternatively, if both parties have a common symmetric encryption system, key confirmation can be obtained by replacing KT_{B1} with $KT_{B1} = F(r_B, g) || E_{KAB}(S_B(DB_1))$, where *E* is a suitable symmetric encryption function.

Key Token Processing and Key Construction (A2) Entity *A* verifies entity *B*'s signature on the key token KT_{B1} using entity *B*'s public verification key, and then verifies entity *A*'s distinguishing identifier and the value $F(r_A, g)$ sent in step (A1). If the checks are 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 $MAC_{KAB}(DB_1)$. Then entity *A* constructs the signed key token

$$KT_{A2} = S_A(DB_2) || MAC_{KAB}(DB_2) || Text5,$$

where $DB_2 = F(r_A, g) || F(r_B, g) || B || Text4$, and sends it to entity *B*.

Key confirmation is provided by including $MAC_{KAB}(DB_2)$ in KT_{A2} . Alternatively, key confirmation can be obtained by replacing KT_{A2} with $KT_{A2} = E_{KAB}(S_A(DB_2))$.

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 checks are successful, entity *B* verifies $MAC_{KAB}(DB_2)$ using $K_{AB} = F(r_B, F(r_A, g))$.

NOTE 1 The number of passes is 3.

NOTE 2 This mechanism provides mutual explicit key authentication and mutual entity authentication.

NOTE 3 This mechanism provides mutual key confirmation.

NOTE 4 This mechanism 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 learn $F(r_A, g)$ prior to choosing the value r_B , entity *B* can 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} .

NOTE 5 Examples of this mechanism (known as the Diffie-Hellman scheme) can be constructed by combining the examples in Annex E.9 with use of a digital signature scheme, such as one of those specified in ISO/IEC 9796 and ISO/IEC 14888.

NOTE 6 This mechanism conforms to ISO/IEC 9798-3. [6] KT_{A1} , KT_{B1} , and KT_{A2} are identical to the tokens sent in the three pass authentication mechanism. The TVPs are also identical, with the following changes of use: the TVP R_A is set to the value $F(r_A, g)$; and the TVP R_B is set to the value $F(r_B, g)$.

NOTE 7 If the data fields Text1 and Text3 (or Text5 and Text3) contain the public key certificates of entities *A* and *B*, respectively, then the second requirement at the beginning of 11.7 can be relaxed to the requirement that all entities are in possession of an authenticated copy of the CA's public verification key.

NOTE 8 If a signature mechanism with text hashing is used, then $F(r_A, g)$ and/or $F(r_B, g)$ need not be sent in key token KT_{B1} . Similarly, neither $F(r_A, g)$ nor $F(r_B, g)$ need be sent in key token KT_{A2} . However, care shall be taken that the random numbers are included in the computation of the respective signatures.

NOTE 9 Key confirmation can alternatively be achieved by encrypting part of the signature. In this case, the third requirement at the beginning of 11.7 does not apply.