INTERNATIONAL STANDARD

ISO/IEC 11770-3

Second edition
2008-07-15

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)

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 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

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
Texti	*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*
$\Vert$	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*:

$$\mathrm{M} = 02 \,||\, \mathrm{B} \,||\, \mathrm{A} \,||\, \mathrm{KT_B} \,||\, \mathrm{KT_A} \,||\, \mathrm{P_B} \,||\, \mathrm{P_A} \,||\, \mathrm{Text1},$$

where *0x02* is the message number.

2) Entity *B* calculates

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

Entity *B* then calculates

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

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 $MAC_{K_B}(M)$ to entity *A*.

4) Entity *A* calculates

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

and verifies $MAC_{K_B}(M)$ 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*:

$$M' = 03 \,||\, A \,||\, B \,||\, KT_A \,||\, KT_B \,||\, P_A \,||\, P_B \,||\, Text2,$$

where *0x03* is the message number.

6) Entity *A* calculates

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

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

7) Entity *A* sends *M'* and $MAC_{K_A}(M')$ to entity *B*.

8) Entity *B* uses K_B to verify $MAC_{K_A}(M')$ 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.

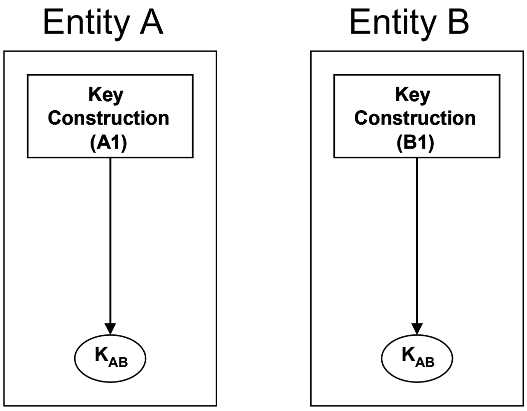


Figure 1: Key Agreement Mechanism 1

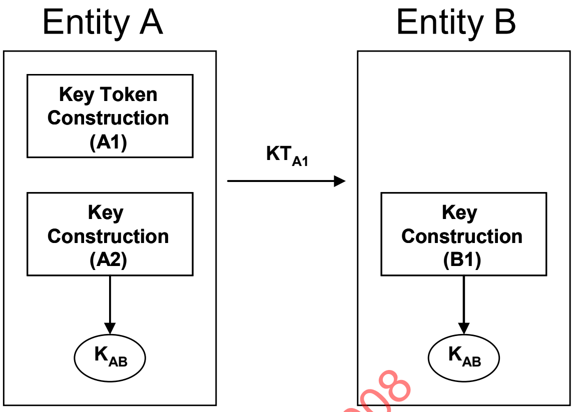


Figure 2: Key Agreement Mechanisms 2, 8

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

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

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.

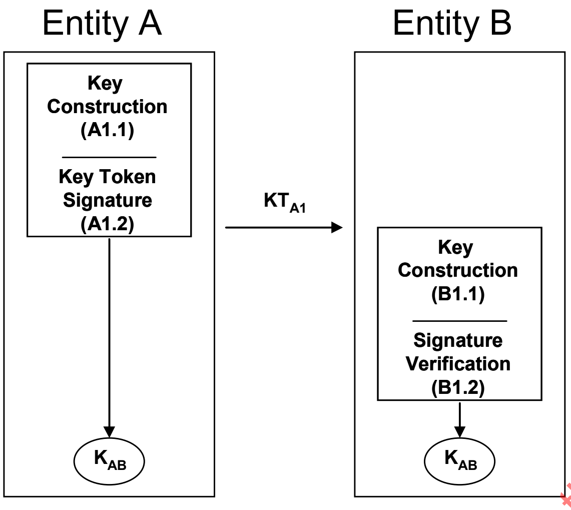


Figure 3: Key Agreement Mechanism 3

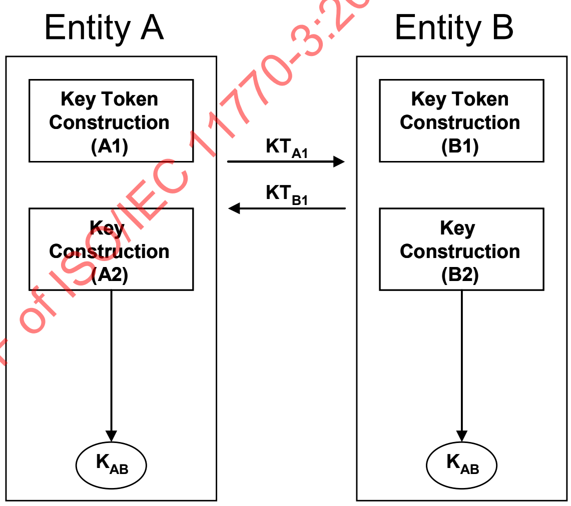


Figure 4: Key Agreement Mechanisms 4, 5, 9

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

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

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

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

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

$$KT_{B1} = F(r_B, g) || Text2$$

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

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

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

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

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-Takashima-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:

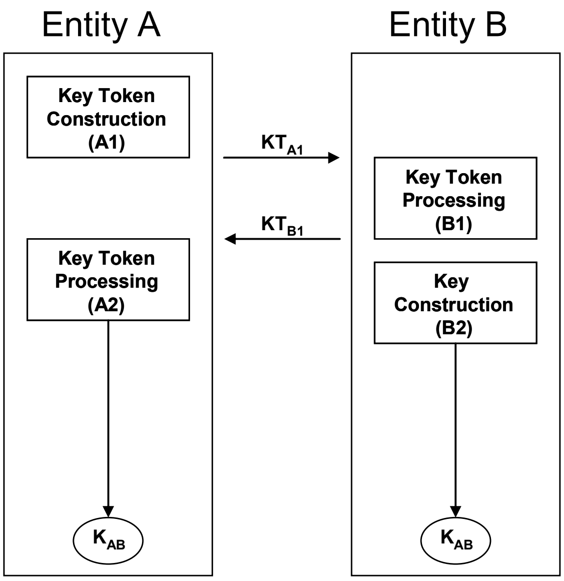
1) Entity *A* has an asymmetric encipherment system with the transformations (E_A, D_A).

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.


Entity A
Entity B
Key Token Construction (A1)
KT_A1
KT_B1
Key Token Processing (B1)
Key Token Processing (A2)
Key Construction (B2)
K_AB
K_AB

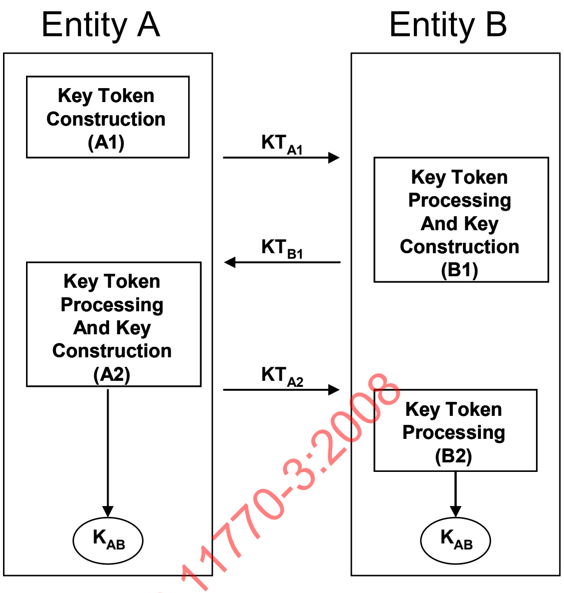

Entity A
Entity B
Key Token Construction (A1)
KT_A1
Key Token Processing And Key Construction (B1)
KT_B1
Key Token Processing And Key Construction (A2)
KT_A2
Key Token Processing (B2)
K_AB
K_AB

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

$$KT_{A1} = r_A \,||\, Text1$$

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

$$BS = S_B(A \,||\, r_A \,||\, r_B \,||\, Text2).$$

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

$$KT_{B1} = E_A(B \,||\, BS \,||\, Text3) \,||\, Text4$$

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* deciphers 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

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

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

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

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 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

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

where

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

and sends it to entity *B*.

Key confirmation is provided by 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.

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. 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 va lues for r_B in the interval between receiving KT_{A1} and sending KT_{B1}.

5) Example: an example of this key agreement mechanism may be provided by the Diffie-Hellman scheme described in Annexes C and D in conjunction with a digital signature scheme such as ISO/IEC 9796.

6) Standards: this mechanism conforms to ISO/IEC 9798-3: Entity authentication using a public key algorithm. KT_{A1}, KT_{B1}, and KT_{A2} are identical to the tokens sent in the three pass authentication mechanism described in subclause 5.2.2 of ISO/IEC 9798-3:1998. Also, the data fields are identical, with the following change of use:

   - the data field R_A (which is present in all three tokens of ISO/IEC 9798-3:1998, subclause 5.2.2) transmits the random function value $F(r_A,g)$

   - the data field R_B (which is present in all three tokens of ISO/IEC 9798-3:1998, subclause 5.2.2) transmits the random function value $F(r_B,g)$

7) Public key certificates: if the data fields *Text1* and *Text3* (or *Text5* and *Text3*) each contain the public key certificates of entities *A* and *B*, respectively, then the requirement 2 at the beginning of this clause can be relaxed to the requirement that all entities are in possession of an authenticated copy of the *CA*'s public verification key.

8) Signature transformation: 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 to be sent in key token KT_{A2}. However, care must be taken that the random numbers are included in the computation of the respective signatures.

9) An alternative is offered for key confirmation by encrypting part of the signature. When the option of encrypting is used instead of a MAC, then requirement 3 does not apply.

10.8 Key agreement mechanism 8

This key agreement mechanism uses elliptic curve cryptography and establishes in one pass 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 2 for a diagram of key agreement mechanism 8.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7. A function is also required to represent the point *P* as an integer. An example function is $\pi(P) = (P_X \bmod 2^{\lceil \rho/2 \rceil}) + 2^{\lceil \rho/2 \rceil}$ where $\rho = \lceil \log_2 n \rceil$ and P_X is the *x*-coordinate of the point *P*.

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

$$KT_{A1} = F(r_A, g)$$

and sends it to entity *B*.

Key construction (A2) Entity *A* computes the shared key

$$K_{AB} = ((r_A + \pi(KT_{A1})\, h_A) \cdot l)\, (\, j \cdot (p_B + \pi(p_B)\, p_B)\,).$$

Key construction (B1) Entity *B* computes the shared key

$$K_{AB} = ((h_B + \pi(p_B)\, h_B) \cdot l)\, (\, j \cdot (KT_{A1} + \pi(KT_{A1})\, p_A)).$$

NOTES – This key agreement mechanism has the following properties:

1) Number of passes: 1

2) Key Authentication: this mechanism provides mutual implicit key authentication.

3) This key agreement mechanism is an example of *MQV* as found in Annex D.10.

10.9 Key agreement mechanism 9

This key agreement mechanism uses elliptic curve cryptography and establishes in two passes 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 4 for a diagram of key agreement mechanism 9.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7. A function is also required to represent the point *P* as an integer. An example function is $\pi(P) = (P_X \bmod 2^{\lceil \rho/2 \rceil}) + 2^{\lceil \rho/2 \rceil}$ where $\rho = \lceil \log_2 n \rceil$ and P_X is the *x*-coordinate of the point *P*.

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

$$KT_{A1} = F(r_A, g)$$

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

$$KT_{B1} = F(r_B, g)$$

and sends it to entity *A*.

Key construction (A2) Entity *A* computes the shared secret key as

$$K_{AB} = ((r_A + \pi(KT_{A1}) h_A) \cdot l)\, (j \cdot (KT_{B1} + \pi(KT_{B1}) p_B)).$$

Key construction (B2) Entity *B* computes the shared secret key as

$$K_{AB} = ((r_B + \pi(KT_{B1}) h_B) \cdot l)\, (j \cdot (KT_{A1} + \pi(KT_{A1}) p_A)).$$

NOTES – This key agreement mechanism has the following properties:

1) Number of passes: *2.*

2) Key Authentication: this mechanism provides mutual implicit key authentication.

3) This key agreement mechanism is an example of MQV with two passes as found in Annex D.11.

10.10 Key agreement mechanism 10

This key agreement mechanism uses elliptic curve cryptography and establishes in three passes 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 7 for a diagram of key agreement mechanism 10.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7. A function is also required to represent the point *P* as an integer. An example function is $\pi(P) = (P_X \bmod 2^{\lceil \rho/2 \rceil}) + 2^{\lceil \rho/2 \rceil}$ where $\rho = \lceil \log_2 n \rceil$ and P_X is the *x*-coordinate of the point *P*.

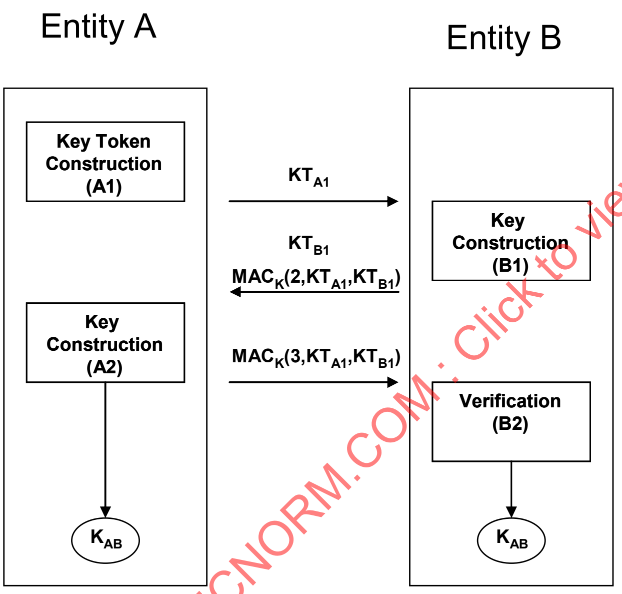


Figure 7: Key Agreement Mechanism 10

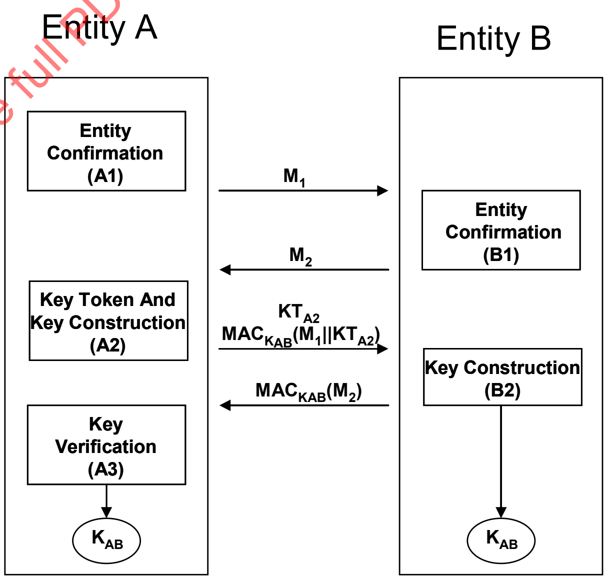


Figure 8: Key Agreement Mechanism 11

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

$$KT_{A1} = F(r_A, g)$$

and sends it to entity *B*.

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

$$KT_{B1} = F(r_B, g).$$

Entity *B* computes the shared secret key as

$$K_{AB} = ((r_B + \pi(KT_{B1})h_B) \cdot l)(j \cdot (KT_{A1} + \pi(KT_{A1})p_A)).$$

Entity *B* then computes the key *K* = *kdf*(K_{AB}). Entity *B* further constructs

$$MAC_K(2 || KT_{A1} || KT_{B1})$$

where *0x02* is the message number, and sends KT_{B1} and $MAC_K(2 || KT_{A1} || KT_{B1})$ to entity *A*.

Key construction (A2) Entity *A* computes the shared secret key as

$$K_{AB} = ((r_A + \pi(KT_{A1})h_A) \cdot l)(j \cdot (KT_{B1} + \pi(KT_{B1})p_B)).$$

Entity *A* computes the key *K* = *kdf*(K_{AB}). Entity *A* computes

$$MAC_K(2 || KT_{A1} || KT_{B1})$$

and verifies what was sent by entity *B*. Entity *A* then computes

$$MAC_K(3 || KT_{A1} || KT_{B1})$$

where *0x03* is the message number, and sends it to entity *B*.

Verification (B2) Entity B computes

$$MAC_K(3 || KT_{A1} || KT_{B1})$$

and verifies entity *A*.

NOTES – This key agreement mechanism has the following properties:

1) Number of passes: *3.*

2) Key Authentication: this mechanism provides mutual explicit key authentication.

3) This key agreement mechanism is an example of MQV with three passes as found in Annex D.12.

10.11 Key agreement mechanism 11

This key agreement mechanism establishes in 4 passes a shared key between entities *A* and *B*. The following requirements shall be satisfied:

1) Each entity *X* has an asymmetric encipherment system with the transformation (E_X, D_X).

2) Each entity has access to an authenticated copy of the public verification transformation of the other entity. This may be achieved by using the mechanisms of Clause 12.

3) Both entities have agreed on a common key derivation function *kdf*.

See Figure 8 for a diagram of key agreement mechanism 11.

Entity Confirmation (A1): Entity *A* sends a message M_1 to Entity *B*, which consists of a random integer r_A and *Text1*.

$$M_1 = (r_A \,||\, \text{Text1}).$$

Entity Confirmation (B1): Entity *B* chooses a random integer r_B, and sends its certificate and *Text2* to entity *A*

$$M_2 = (r_B \,||\, \text{Cert}_B \,||\, \text{Text2}).$$

Key Token and Key Construction (A2): Entity *A* extracts entity *B*'s public key from the certificate and verifies the certificate. Entity *A* then generates a random integer r_A' and computes the shared key

$$K_{AB} = \text{kdf}(r_A, r_B, r_A').$$

Entity *A* then sends the encrypted key token

$$KT_{A2} = E_B(r_A')$$

and

$$MAC_{K_{AB}}(M_1 \,||\, KT_{A2})$$

to Entity *B*.

Key Construction (B2): Entity *B* decrypts KT_{A2} and computes the shared key

$$K_{AB} = \text{kdf}(r_A, r_B, r_A').$$

Entity *B* computes

$$MAC_{K_{AB}}(M_1 \,||\, KT_{A2})$$

and compares it to the received MAC value. Entity *B* sends

$$MAC_{K_{AB}}(M_2)$$

to Entity *A*.

Key Verification (A3): Entity *A* computes $MAC_{K_{AB}}(M_2)$ and compares it to the received MAC value.

NOTES – This key agreement mechanism has the following properties:

1) Number of passes: 4.

2) Key Authentication: this mechanism provides mutual explicit key authentication.

3) This key agreement is derived from the Transport Layer Security (TLS) protocol [12]. TLS can be regarded as an example of this mechanism. The key agreement is known as the TLS handshake phase.

4) TLS has a cipher suite which is a list of encryption algorithms that is supported by an entity. *Text1* and *Text2* are used for the cipher suite negotiation in SSL by entities *A* and *B* respectively.

11 Secret key transport

In this part of ISO/IEC 11770, 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 techniques.

> NOTE In practical implementations of the key transport mechanisms, the key data block may be subject to further processing prior to being used for encipherment. For instance, the key data block may be xored by a (pseudo) random bit pattern to destroy any apparent structure in the key data block. Obtaining key material for the key *K* is beyond the scope of this part of ISO/IEC 11770. Refer to ISO/IEC 18031 for the generation of random bits.

11.1 Key transport mechanism 1

This key transport mechanism transfers in one pass a secret key from entity *A* to entity *B* with implicit key authentication from entity *B* to entity *A*. The following requirements shall be satisfied:

1) Entity *B* has an asymmetric encipherment system (E_B,D_B).

2) Entity *A* has access to an authenticated copy of entity *B*'s public encipherment transformation E_B. This may be achieved using the mechanisms of Clause 12.

3) The optional *TVP* shall either be a time stamp or sequence number. If time stamps are used, then the entities *A* and *B* need to maintain synchronous clocks or use a Trusted Third Party Time Stamp Authority. If sequence numbers are used, then entities *A* and *B* have to maintain bilateral counters.

See Figure 9 for a diagram of key transport mechanism 1.

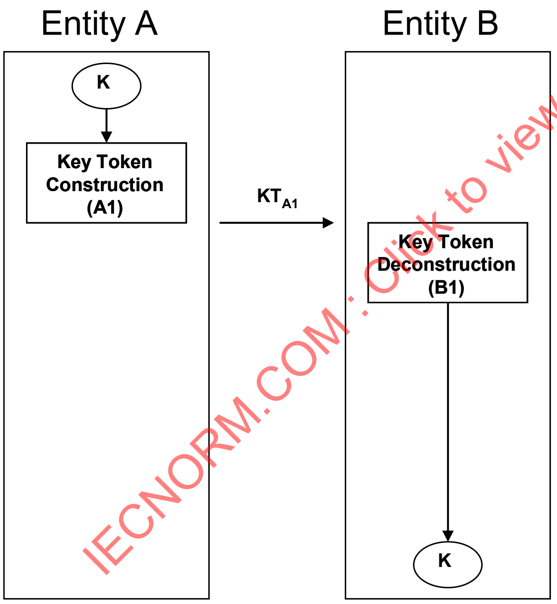


Figure 9: Key Transport Mechanism 1

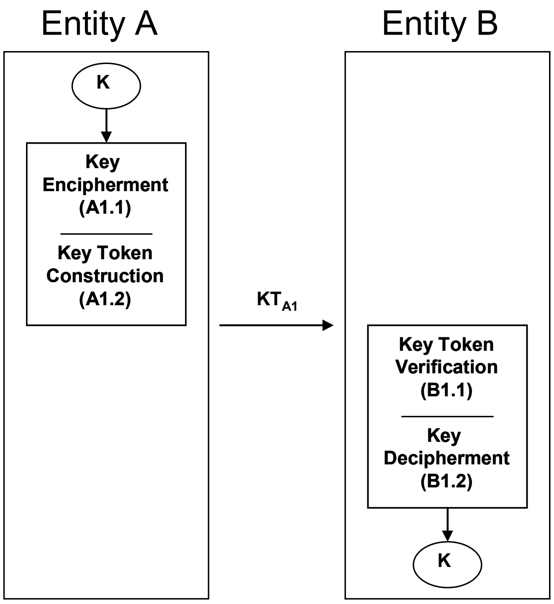


Figure 10: Key Transport Mechanism 2

Key Token Construction (A1) Entity *A* has obtained a key *K* and wants to transfer it securely to entity *B*. Entity *A* constructs a key data block consisting of its distinguishing identifier for entity *A* (optional), the key *K*, an optional *TVP* and an optional data field *Text1*. Then entity *A* encrypts the key data block using the receiver's public encipherment transformation E_B and sends the key token

$$KT_{A1} = E_B(A \,||\, K \,||\, TVP \,||\, Text1) \,||\, Text2$$

to entity *B*.

Key Token Deconstruction (B1) Entity *B* deciphers the received key token KT_{A1} using its private decipherment transformation D_B, recovers the key *K*, checks the optional *TVP*, and associates the recovered key *K* with the claimed originator entity *A*.

NOTES - This Key Transport 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* since only entity *B* can possibly recover the key *K*.

3) Key confirmation: this mechanism provides no key confirmation.

4) Key control: *A* can choose the key.

5) *TVP*: the optional *TVP* prevents the replay of the key token.

6) Key usage: as entity *B* receives the key *K* from the non-authenticated entity *A*, secure usage of *K* by entity *B* is restricted to functions not requiring trust in entity *A*'s authenticity, such as decipherment and generation of message authentication codes.

7) Example: an example of this mechanism (known as ElGamal key transfer) is described in Annex E.1. Another example of this mechanism using RSA is described in Annex E.3.

11.2 Key transport mechanism 2

This key transport mechanism is an extension of the one-pass entity authentication mechanism in ISO/IEC 9798-3. It transfers a secret key enciphered and signed from entity *A* to entity *B* with explicit key authentication from entity *A* to entity *B* and implicit key authentication from entity *B* to entity *A*. The following requirements shall be satisfied:

1) Entity *A* has an asymmetric signature system (S_A, V_A).

2) Entity *B* has an asymmetric encipherment system (E_B, D_B).

3) Entity *A* has access to an authenticated copy of entity *B*'s public encipherment transformation E_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 verification transformation V_A. This may be achieved using the mechanisms of Clause 12.

5) The optional *TVP* shall either be a time stamp or sequence number. If time stamps are used, then the entities *A* and *B* need to maintain synchronous clocks or use a Trusted Third Party Time Stamp Authority. If sequence numbers are used then entities *A* and *B* have to maintain bilateral counters.

See Figure 10 for a diagram of key transport mechanism 2.

Key Encipherment (A1.1) Entity *A* has obtained a key *K* and wants to transfer it securely to entity *B*. Entity *A* forms the key data block, consisting of the sender's distinguishing identifier for entity *A*, the key *K* and an optional data field *Text1*. Then entity *A* enciphers the key data block with entity *B*'s public encipherment transformation E_B and forms the enciphered block

$$BE = E_B(A \| K \| Text1).$$

Key Token Construction (A1.2) Entity *A* forms the token data block, consisting of the recipient's distinguishing identifier for entity *B*, an optional time stamp or sequence number *TVP*, the enciphered block *BE* and the optional data field *Text2*. Then entity *A* signs the token data block using its private signature transformation S_A and some optional *Text3*, and then sends the resulting key token

$$KT_{A1} = S_A(B \,||\, TVP \,||\, BE \,||\, Text2) \,||\, Text3$$

to entity *B*.

Key Token Verification (B1.1) Entity *B* uses the sender's public verification transformation V_A to verify the digital signature of the received key token KT_{A1}. Then entity *B* checks the receiver identification for entity *B* and optionally the *TVP*.

Key Decipherment (B1.2) Entity *B* deciphers the block *BE* with its private decipherment transformation D_B. Then entity *B* compares the field for entity *A* in block *BE* with the identity of the signing entity. If all checks are successful, entity *B* accepts the key *K*.

NOTES - This Key Transport Mechanism has the following properties:

1) Number of passes: *1*.

2) Key and entity authentication: this mechanism provides entity authentication of entity *A* to entity *B* if the optional *TVP* is used, and implicit key authentication from entity *B* to entity *A*.

3) Key confirmation: from entity *A* to entity *B*. Entity *B* can be sure that it shares the correct key with entity *A*, but entity *A* can only be sure that entity *B* has indeed received the key after it has obtained a positive reply from entity *B* enciphered using key *K*.

4) *TVP* (optional): provides entity authentication of entity *A* to entity *B* and prevents replay of the key token. In order to prevent replay of the key data block *BE*, an additional *TVP* may also be included in *Text1*.

5) Key control: Entity *A* can choose the key K_A, since it is the originating entity. Similarly, entity *B* can choose the key K_B. Joint key control can be achieved by each entity by combining the two keys K_A and K_B on both sides to form a shared secret key K_{AB}. An extra pass is required for joint key control. However, the combination function must be one-way, otherwise entity *A* can choose the shared secret key. This mechanism could then be classified as a key agreement mechanism.

6) Data field for entity *A*: Entity *A*'s distinguishing identifier is included in the enciphered block *BE* to prevent entity *A* from misappropriating an enciphered key block intended for use by another entity. This is achieved by comparing entity *A*'s identity with entity *A*'s signature on the token.

7) Standards: conformance with ISO/IEC 9798-3: Entity authentication using a public key algorithm. KT_{A1} is compatible to the token sent in the one-pass authentication mechanism described in subclause 5.1.1 of ISO/IEC 9798-3:1998. The token accommodates the transfer of the key *K* through use of the optional text field: *Text1* has been replaced by *BE* || *Text2*.

8) Public key certificates: the data field *Text3* may be used to deliver the public key certificate of entity *A*. Then the requirement 4 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.

9) Mutual entity authentication and joint key control: if two executions of this key transport mechanism are combined (from entity *A* to entity *B* and from entity *B* to entity *A*), then mutual entity authentication and joint key control can be provided (depending on the use of the optional *TVP*).

10) Usage: Key transport mechanism 2 is intended to be used in environments where confidentiality of parts of a message is needed, e.g. a message that carries many confidential elements as well as the enciphered keys.

11) Examples of this mechanism are described in Annexes E.2 and E.5.

11.3 Key transport mechanism 3

This key transport mechanism transfers in one pass a secret key signed and enciphered from entity *A* to entity *B* with unilateral key confirmation. The following requirements shall be satisfied:

1) Entity *A* has an asymmetric signature system (S_A, V_A).

2) Entity *B* has an asymmetric encipherment system (E_B, D_B).

3) Entity *A* has access to an authenticated copy of entity *B*'s public encipherment transformation E_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 verification transformation V_A. This may be achieved using the mechanisms of Clause 12.

5) The optional *TVP* shall either be a time stamp or a sequence number: If time stamps are used then the entities *A* and *B* need to maintain synchronous clocks. If sequence numbers are used then entities *A* and *B* have to maintain bilateral counters.

See Figure 11 for a diagram of key transport mechanism 3.

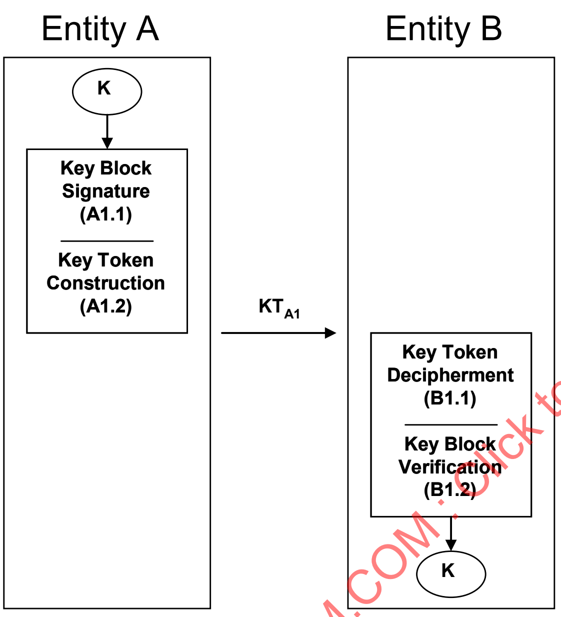

Entity A
Entity B
K
Key Block Signature (A1.1)
Key Token Construction (A1.2)
KT$_{A1}$
Key Token Decipherment (B1.1)
Key Block Verification (B1.2)
K

Figure 11: Key Transport Mechanism 3

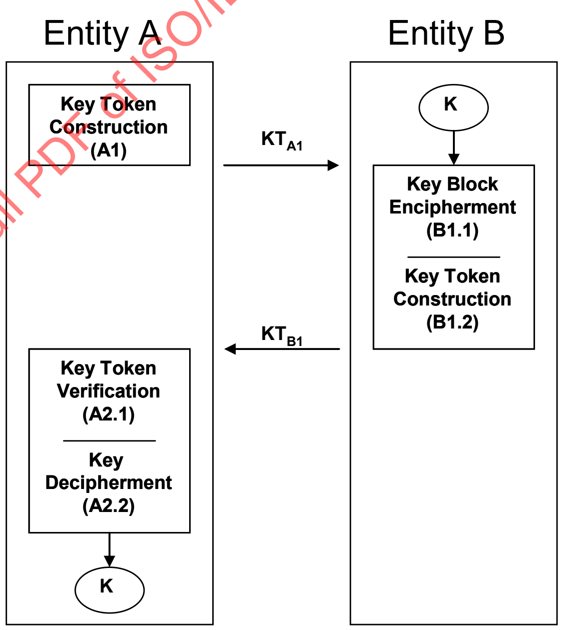

Entity A
Entity B
Key Token Construction (A1)
KT$_{A1}$
K
Key Block Encipherment (B1.1)
Key Token Construction (B1.2)
KT$_{B1}$
Key Token Verification (A2.1)
Key Decipherment (A2.2)
K

Figure 12: Key Transport Mechanism 4

Key Block Signature (A1.1) Entity *A* has obtained a key *K* and wants to transfer it securely to entity *B*. Entity *A* forms a key data block consisting of the recipient's distinguishing identifier for entity *B*, the key *K*, an optional sequence number or time stamp *TVP*, and some optional data. Then entity *A* signs the key block using its private signature transformation S_A to generate the signed block

$$BS = S_A(B || K || TVP || Text1).$$

Key Token Construction (A1.2) Entity *A* forms the token data block, consisting of the signed block *BS* and some optional *Text2*. Then entity *A* enciphers the token data block using the receiver's public encipherment transformation E_B and some optional *Text3*, and then sends the resulting key token

$$\mathrm{KT}_{A1} = \mathrm{E}_{B}(\mathrm{BS} \,||\, \mathrm{Text2}) \,||\, \mathrm{Text3}$$

to entity *B*.

Key Token Decipherment (B1.1) Entity *B* deciphers the received key token KT_{A1} using its private decipherment transformation D_B.

Key Block Verification (B1.2) Entity *B* uses the sender's public verification transformation V_A to verify the integrity and origin of *BS*. Entity *B* validates that it is the intended recipient of the token (by inspection of the identifier for entity *B*) and, optionally, that the token has been sent timely (by inspection of *TVP*). If all verifications are successful, entity *B* accepts the key *K*.

NOTES - This Key Transport Mechanism has the following properties:

1) Number of protocol passes: 1.

2) Key and entity authentication: this mechanism provides entity authentication of entity *A* to entity *B* if the optional *TVP* is used, and implicit key authentication from entity *B* to entity *A*.

3) Key confirmation: from entity *A* to entity *B*. Entity *B* can be sure that it shares the correct key *K* with entity *A*, but entity *A* can only be sure that entity *B* has indeed received the key after it has obtained a positive reply from entity *B* enciphered using key *K*.

4) Key control: Entity *A* can choose the key.

5) *TVP* (optional): may provide entity authentication of entity *A* to entity *B* and prevent replay of the key token.

6) Data field for entity *B*: Entity *B*'s distinguishing identifier is included in the signed key block *BS* to explicitly indicate the recipient of the key, thereby preventing misuse of the signed block *BS* by entity *B*.

7) Public key certificates: the data field *Text3* may be used to deliver the public key certificate of entity *A*. Then the requirement 4 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.

8) Mutual entity authentication and joint key control: if two executions of this key transport mechanism are combined (from entity *A* to entity *B* and from entity *B* to entity *A*) then mutual entity authentication and joint key control can be provided (depending on the use of the optional *TVP*).

11.4 Key transport mechanism 4

This key transport mechanism is based on the two-pass authentication mechanism of ISO/IEC 9798-3 and transfers a key from entity *B* to entity *A*. The following requirements shall be satisfied:

1) Entity *A* has an asymmetric encipherment system (E_A, D_A).

2) Entity *B* has an asymmetric signature system (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 12 for a diagram of key transport mechanism 4.

Key Token Construction (A1) Entity *A* constructs the key token KT_{A1}, consisting of a random number r_A and an optional data field *Text1*,

$$KT_{A1} = r_A \,||\, Text1$$

and sends it to entity *B*.

Key Block Encipherment (B1.1) Entity *B* has obtained a key *K* and wants to transfer it securely to entity *A*. Entity *B* forms a key data block, consisting of the sender's distinguishing identifier for entity *B*, the key *K* and an optional data field *Text2*. Then entity *B* enciphers the key data block with entity *A*'s public encipherment transformation E_A and forms the enciphered block

$$BE = E_A(B \,||\, K \,||\, Text2)$$

Key Token Construction (B1.2) Entity *B* forms the token data block, consisting of the recipient's distinguishing identifier for entity *A*, the random number r_A received in step (A1), the new random number r_B (optional), the enciphered block *BE*, and the optional data field *Text3*. Then entity *B* signs the token data block with its private signature transformation S_B and some optional *Text4*, and then sends the resulting key token

$$KT_{B1} = S_B(A \,||\, r_A \,||\, r_B \,||\, BE \,||\, Text3) \,||\, Text4$$

to entity *A*.

Key Token Verification (A2.1) Entity *A* uses the sender's public verification transformation V_B to verify the digital signature of the received key token KT_{B1}. Then entity *A* checks the distinguishing identifier for entity *A* and checks that the received value r_A agrees with the random number sent in step (A1).

Key Block Decipherment (A2.2) Entity *A* deciphers the block *BE* with its private decipherment transformation D_A. Then entity *A* validates the sender's distinguishing identifier for entity *B*. If all checks are successful, entity *A* accepts the key *K*.

NOTES - This Key Transport Mechanism has the following properties:

1) Number of protocol passes: 2.

2) Key and entity authentication: this mechanism provides entity authentication of entity *B* to entity *A* and implicit key authentication from entity *A* to entity *B*.

3) Key confirmation: from entity *B* to entity *A*. Entity *A* can be sure that it shares the correct key *K* with entity *B*, but entity *B* can only be sure that entity *A* has indeed received the key after it has obtained a secured message from entity *A* which has been unambiguously processed.

4) Key control: Entity *B* can choose the key.

5) Standards: conformance with ISO/IEC 9798-3 Entity authentication using a public key algorithm. The tokens KT_{A1} and KT_{B1} are compatible with the tokens sent in the two-pass authentication mechanism described in subclause 5.1.2 of ISO/IEC 9798-3:1998 (note that the roles of entities *A* and *B* are exchanged). The token KT_{B1} accommodates the transfer of the key *K* through use of the optional data field: *Text2* has been replaced by *BE* || *Text3*.

6) Standards: if this key transport mechanism is executed twice in parallel between two entities, then the resulting mutual key transport mechanism is in conformance with the mechanism described in subclause 5.2.3. Two pass parallel authentication of ISO/IEC 9798-3.

7) Data field r_B is shown for consistency with ISO/IEC 9798-3. Because of the presence of *BE* in KT_{B1} the data field r_B is no longer required and is therefore optional in this mechanism.

8) Mutual entity authentication and joint key control: if two executions of this key transport mechanism are combined (from entity *A* to entity *B* and from entity *B* to entity *A*), then mutual entity authentication and joint key control can be provided.

11.5 Key transport mechanism 5

This key transport mechanism is based on the three-pass authentication mechanism of ISO/IEC 9798-3 and transfers in three passes two shared secret keys with mutual entity authentication and key confirmation. One key is transferred from entity *A* to entity *B* and one key from entity *B* to entity *A*. The following requirements shall be satisfied:

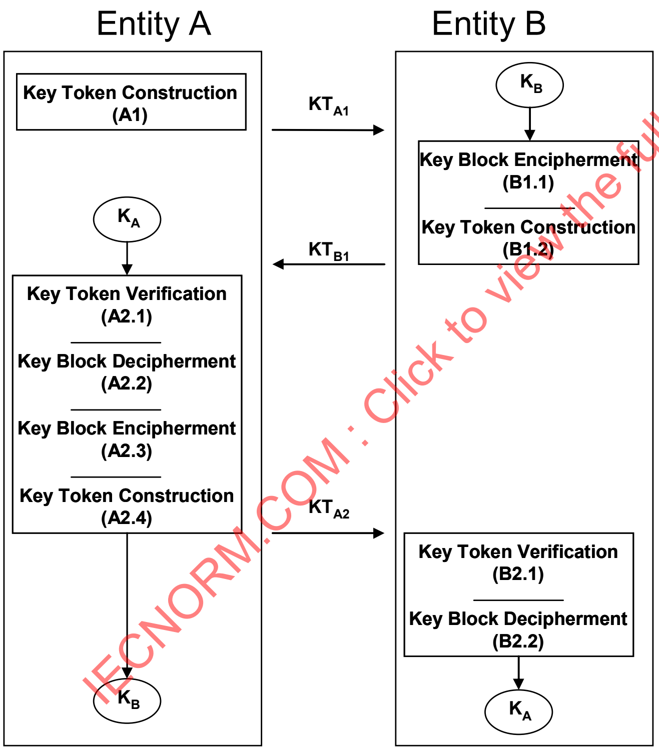
1) Each entity *X* has an asymmetric signature system (S_X, V_X).

2) Each entity *X* has an asymmetric encipherment system (E_X, D_X).

3) 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.

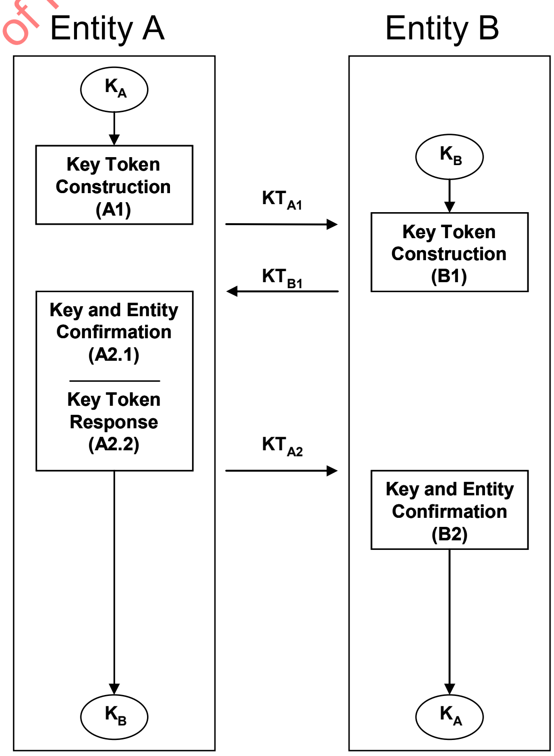
4) Each entity has access to an authenticated copy of the public encipherment transformation of the other entity. This may be achieved using the mechanisms of Clause 12.

See Figure 13 for a diagram of key transport mechanism 5



Entity A
Entity B
Key Token Construction (A1)
KT_{A1}
K_B
Key Block Encipherment (B1.1)
Key Token Construction (B1.2)
K_A
KT_{B1}
Key Token Verification (A2.1)
Key Block Decipherment (A2.2)
Key Block Encipherment (A2.3)
Key Token Construction (A2.4)
KT_{A2}
Key Token Verification (B2.1)
Key Block Decipherment (B2.2)
K_B
K_A

Figure 13: Key Transport Mechanism 5



Entity A
Entity B
K_A
Key Token Construction (A1)
KT_{A1}
K_B
Key Token Construction (B1)
KT_{B1}
Key and Entity Confirmation (A2.1)
Key Token Response (A2.2)
KT_{A2}
Key and Entity Confirmation (B2)
K_B
K_A

Figure 14: Key Transport Mechanism 6

Key Token Construction (A1) Entity *A* randomly generates r_A and constructs the key token

$$KT_{A1} = r_A \,||\, \text{Text1}$$

and sends it to entity *B*.

Key Block Encipherment (B1.1) Entity *B* has obtained a key K_B and wants to transfer it securely to entity *A*. Entity *B* constructs a block containing its own distinguishing identifier for entity *B*, the key K_B, and some optional *Text2*, and enciphers the block, using the recipient's public encipherment transformation E_A

$$BE_1 = E_A(B \,||\, K_B \,||\, Text2).$$

Key Token Construction (B1.2) Then entity *B* randomly generates r_B and constructs a data block, containing r_B, r_A, the recipient's identity for entity *A*, the enciphered key block BE_1, and some optional *Text3*. Entity *B* signs the block using its private signature transformation S_B and some optional *Text4*, and then sends the key token

$$KT_{B1} = S_B(r_B \,||\, r_A \,||\, A \,||\, BE_1 \,||\, Text3) \,||\, Text4$$

to entity *A*.

Key Token Verification (A2.1) Entity *A* verifies entity *B*'s signature on the key token KT_{B1} using entity *B*'s public verification transformation V_B, checks the distinguishing identifier for entity *A* and checks that the received value r_A agrees with the random number sent in step (A1).

Key Block Decipherment (A2.2) Entity *A* deciphers the enciphered block BE_1 using its private decipherment transformation D_A and checks the distinguishing identifier for entity *B*. If all checks are successful, entity *A* accepts the key K_B.

Key Block Encipherment (A2.3) Then entity *A* constructs a data block, containing its own distinguishing identifier for entity *A*, its own key K_A, and some optional *Text5*, and enciphers the block, using the recipient's public encipherment transformation E_B

$$BE_2 = E_B(A \,||\, K_A \,||\, Text5).$$

Key Token Construction (A2.4) Then entity *A* constructs a data block, containing the random number r_A, the random number r_B, the recipient's distinguishing identifier for entity *B*, the enciphered key block BE_2, and some optional *Text6*. Entity *A* signs the data block using its private signature transformation S_A and some optional *Text7*, and then sends the key token

$$KT_{A2} = S_A(r_A \,||\, r_B \,||\, B \,||\, BE_2 \,||\, Text6) \,||\, Text7$$

to entity *B*.

Key Token Verification (B2.1) Entity *B* verifies entity *A*'s signature on the key token KT_{A2}, using entity *A*'s public verification transformation V_A, checks the distinguishing identifier for entity *B* and checks that the received value r_B agrees with the random number sent in step (B1.2). In addition, *B* checks that the received value r_A agrees with the one contained in KT_{A1}.

Key Block Decipherment (B2.2) Entity *B* deciphers the enciphered block BE_2 using its private decipherment transformation D_B and verifies the distinguishing identifier for entity *A*. If all checks are successful, entity *B* accepts the key K_A. If only unilateral key transport is required then as appropriate either BE_1 or BE_2 can be omitted.

NOTES - This Key Transport Mechanism has the following properties:

1) Number of passes: *3*.

2) Key and entity authentication: this mechanism provides mutual entity authentication, implicit key authentication of K_A from entity *B* to entity *A* and implicit key authentication of K_B from entity *A* to entity *B*.

3) Key confirmation: this mechanism provides key confirmation from sender to recipient for both keys K_A and K_B. Moreover, if entity *A* includes a *MAC* on K_B in the data field *Text6* of KT_{A2}, then this mechanism provides mutual key confirmation with respect to K_B.

4) Key control: Entity *A* can choose the key K_A, since it is the originating entity. Similarly, entity *B* can choose the key K_B. Joint key control can be achieved by each entity by combining the two keys K_A and K_B on both sides to form a shared secret key K_{AB}. However, the combination function must be one-way, otherwise entity *A* can choose the shared secret key. This mechanism could then be classified as a key agreement mechanism.

5) Standards: conformance to ISO/IEC 9798-3, KT_{A1}, KT_{B1}, and KT_{A2} are compatible to the tokens sent in the three pass authentication mechanism described in Clause 5.2.2 of ISO/IEC 9798-3:1998. The second token accommodates the transfer of the key K_B: *Text2* has been replaced by BE_1||*Text3*. The third token accommodates the transfer of the key K_A: *Text4* has been replaced by BE_2||*Text6*. The third token may also accommodate the transfer of a *MAC* within *Text6*.

6) Public key certificates: if the data fields *Text1* and *Text4* (or *Text7* and *Text4*) each contain the public key certificates of entities *A* and *B*, respectively, then the requirement 3 and 4 at the beginning of this clause can be relaxed to the requirement that all entities are in possession of an authenticated copy of the *CA*'s public verification key.

7) Signature transformation: if a signature mechanism with text hashing is used, then optionally the random number r_A need not be sent in the key token KT_{B1}. Analogously, neither r_A nor r_B need to be sent in key token KT_{A2}. However, care must be taken that the random numbers are included in the computation of the respective signatures.

11.6 Key transport mechanism 6

This key transport mechanism securely transfers in three passes two secret keys, one from entity *A* to entity *B* and one from entity *B* to entity *A*. In addition, the mechanism provides mutual entity authentication and mutual key confirmation about their respective keys. This mechanism is based on the following requirements:

1) Each entity *X* has an asymmetric encipherment system (E_X,D_X).

2) Each entity has access to an authenticated copy of the public encipherment transformation of the other entity. This may be achieved using the mechanisms of Clause 12.

See Figure 14 for a diagram of key transport mechanism 6.

Key Token Construction (A1) Entity *A* has obtained a key K_A and wants to transfer it securely to entity *B*. Entity *A* selects a random number r_A and constructs a key data block consisting of its distinguishing identifier, the key K_A, the number r_A and an optional data field *Text1*. Then entity *A* enciphers the key block using entity *B*'s public encipherment transformation E_B, thereby producing the enciphered data block

$$BE_1 = E_B(A \,||\, K_A \,||\, r_A \,||\, Text1).$$

Entity *A* constructs the token KT_{A1}, consisting of the enciphered data block and some optional data field *Text2*

$$KT_{A1} = BE_1 \,||\, Text2.$$

Entity *A* sends the token to entity *B*.

Key Token Construction (B1) Entity *B* extracts the enciphered key block BE_1 from the received key token KT_{A1} and deciphers it using its private decipherment transformation D_B. Then entity *B* checks that the decrypted version of BE_1 contains the identifier for entity *A*.

Entity *B* has obtained a key K_B and wants to transfer it securely to entity *A*. Entity *B* selects a random number r_B and constructs a key data block consisting of the distinguishing identifier for entity *B*, the key K_B, the random number r_B, the random number r_A (as extracted from the deciphered block) and an optional data field *Text3*. Then entity *B* enciphers the key block using entity *A*'s public encipherment transformation E_A, thereby producing the enciphered data block

$$BE_2 = E_A(B || K_B || r_A || r_B || Text3).$$

Then entity *B* constructs the key token KT_{B1}, consisting of the enciphered data block BE_2 and an optional data field *Text4*,

$$KT_{B1} = BE_2 || Text4.$$

Entity *B* sends the token to entity *A*.

Key and Entity Confirmation (A2.1) Entity *A* extracts the enciphered key block BE_2 from the received key token KT_{B1} and deciphers it using its private decipherment transformation D_A. Then entity *A* checks the validity of the key token through comparison of the random number r_A with the random number r_A contained in the enciphered block BE_2. If the verification is successful, entity *A* has authenticated entity *B* and at the same time obtained confirmation that K_A has safely reached entity *B*.

Key Token Response (A2.2) Entity *A* extracts the random number r_B from the deciphered key block and constructs the key token KT_{A2}, consisting of the random number r_B and an optional data field *Text5*

$$KT_{A2} = r_B || Text5.$$

Entity *A* sends the token to entity *B*.

Key and Entity Confirmation (B2) Entity *B* verifies that the response r_B extracted from KT_{A2} is consistent with the random number r_B sent in enciphered form in KT_{B1}. If the verification is successful, entity *B* has authenticated entity *A* and at the same time has obtained confirmation that K_B has safely reached entity *A*.

NOTES - This Key Transport Mechanism has the following properties:

1) Number of passes: *3*.

2) Entity authentication: this mechanism provides mutual entity authentication, implicit key authentication of K_A from entity *B* to entity *A* and implicit key authentication of K_B from entity *A* to entity *B*.

3) Key confirmation: this mechanism provides mutual key confirmation.

4) Key control: Entity *A* can choose the key K_A, since it is the originating entity. Similarly, entity *B* can choose the key K_B. Joint key control can be achieved by each entity by combining the two keys K_A and K_B on both sides to form a shared secret key K_{AB}. However, the combination function must be one-way, otherwise entity *B* can choose the shared secret key. This mechanism could then be classified as a key agreement mechanism.

5) Key usage: this mechanism uses asymmetric techniques to mutually transfer two secret keys, K_A from entity *A* to entity *B* and K_B from entity *B* to entity *A*. The following cryptographic function separation may be derived from the mechanism: entity *A* uses its key K_A to encipher messages for entity *B* and to verify authentication codes from entity *B*. Entity *B* in turn uses the received key K_A to decipher messages from entity *A* and generate authentication codes for entity *A*. The cryptographic functions of K_B may be separated in an analogous manner. In such a way, the asymmetric basis of the key transport mechanism may be extended to the usage of the secret keys.

6) Example: this mechanism is derived from the three pass protocol known as COMSET [7].

7) Background: this mechanism is based on zero-knowledge techniques. From the execution of the mechanism, neither of the entities learns anything that it could not have computed itself.

12 Public key transport

This clause describes key management mechanisms that make an entity's public key available to other entities in an authenticated fashion. Authenticated distribution of public keys is an essential security requirement. This authenticated distribution can be achieved in different ways:

1) Public key distribution without a trusted third party.

2) 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.

12.1 Public key distribution without a trusted third party

This subclause describes mechanisms which provide authenticated distribution of public keys without the involvement of a trusted third party.

12.1.1 Public key transport mechanism 1

If entity *A* has access to a protected channel (i.e. a channel which provides data origin authentication and data integrity), such as a courier, registered mail, etc., to entity *B* then entity *A* may transport its public key information directly via that protected channel to entity *B*. This is the most elementary form of transferring a public key. The following requirements shall be satisfied:

1) Entity *A*’s public key information PKI_A contains at least entity *A*'s distinguishing identifier and entity *A*'s public key. In addition it may contain a serial number, a validity period, a time stamp and other data elements.

2) Since the public key information *PKI* does not contain any secret data, the communication channel need not provide confidentiality.

See Figure 15 for a diagram of public key transport mechanism 1.

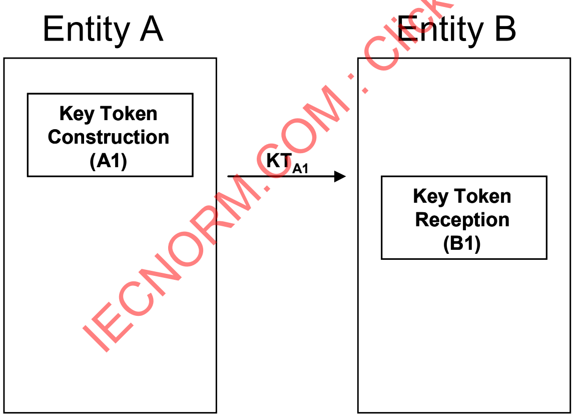


Figure 15: Public Key Transport Mechanism 1

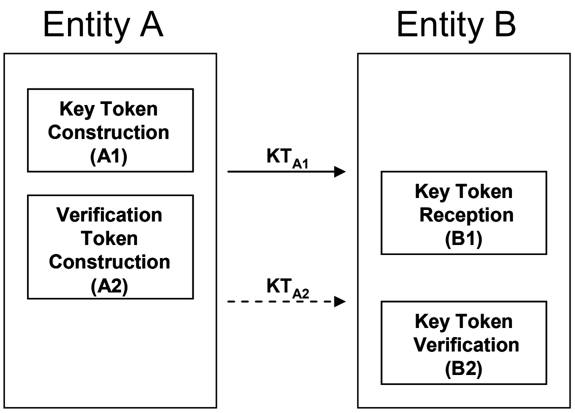

Entity A
Entity B
Key Token Construction (A1)
Verification Token Construction (A2)
KTA1
KTA2
Key Token Reception (B1)
Key Token Verification (B2)

Figure 16: Public Key Transport Mechanism 2

Key Token Construction (A1) Entity *A* constructs the key token KT_{A1} containing the public key information of entity *A* and some optional data field *Text*, and sends it via a protected channel to entity *B*.

$$KT_{A1} = PKI_A \,||\, Text.$$

Key Token Reception (B1) Entity *B* receives the key token via the protected channel from entity *A*, retrieves entity *A*'s public key information PKI_A and stores entity *A*'s public key into the list of active public keys (this list shall be protected from tampering).

NOTES - This Public Key Transport Mechanism has the following properties:

1) This mechanism can be used to transfer public verification keys (for an asymmetric signature system) or public encipherment keys (for an asymmetric encipherment system) or public key agreement keys.

2) Authentication in this context includes both data integrity and data origin authentication (as defined in ISO 7498-2:1989).

12.1.2 Public key transport mechanism 2

This mechanism transports the public key information of entity *A* via an unprotected channel to entity *B*. To verify the integrity and the origin of the received public key information a second authenticated channel is used. Such a mechanism is useful when the public key information *PKI* is transferred electronically on a high bandwidth channel, whereas the authentication of the public key information takes place over an authenticated low bandwidth channel such as a telephone, courier, or registered mail. As an additional requirement, the entities shall share a common hash, as defined in ISO/IEC 10118-1. The following requirements shall be satisfied:

1) Entity *A*’s public key information PKI_A contains at least entity *A*'s distinguishing identifier and entity *A*'s public key. In addition it may contain a serial number, a validity period, a time stamp and other data elements.

2) Since the public key information *PKI* does not contain any secret data, the communication channel need not provide confidentiality.

See Figure 16 for a diagram of public key transport mechanism 2.

Key Token Construction (A1) Entity *A* constructs the key token KT_{A1} containing the public key information of entity *A* and sends it to entity *B*.

$$KT_{A1} = PKI_A \,||\, Text1.$$

Key Token Reception (B1) Entity *B* receives the key token, retrieves entity *A*'s public key information PKI_A, and stores it protected from tampering for later verification and use.

Verification Token Construction (A2) Entity *A* computes a check value $hash(PKI_A)$ on its public key information and sends this check value together with the optional distinguishing identifiers of entities *A* and *B* to entity *B* using a second independent and authenticated channel (e.g. a courier or registered mail).

$$KT_{A2} = A \,||\, B \,||\, hash(PKI_A) \,||\, Text2.$$

Key Token Verification (B2) Upon reception of the verification token KT_{A2}, *B* optionally checks the distinguishing identifier of entities *A* and *B*, computes the check value on the public key information of entity *A* received in the key token KT_{A1} and compares it with the check value received in the verification token KT_{A2}. If the check succeeds, entity *B* puts entity *A*'s public key into the list of active public keys (this list shall be protected from tampering).

NOTES - This Public Key Transport Mechanism has the following properties:

1) This mechanism can be used to transfer public verification keys (for an asymmetric signature system) or public encipherment keys (for an asymmetric encipherment system) or public key agreement keys.

2) Authentication in this context includes both data integrity and data origin authentication.

3) If the public key that is transported is a key for an asymmetric signature system not giving message recovery, then entity *A* may sign the token KT_{A1} using the corresponding private signature key. In that case, the verification of entity *A*'s signature in step (B1) using the received public verification key confirms that entity *A* knew the corresponding private signature key, and so presumably, was the only entity that knew the corresponding private signature key at the time the token was created. If a time stamp is used in *PKI*, then verification confirms that entity *A* currently knows the corresponding private signature key.

4) A manually signed letter from Entity *A* may be used for the verification token.

12.2 Public key distribution using a trusted third party

The authentication of the entities' public keys can be ensured by exchanging the public keys in the form of public key certificates. Entity *A*'s public key certificate contains the public key information, together with a digital signature provided by a trusted third party, the Certification Authority (*CA*). The introduction of a *CA* reduces the problem of authenticated user public key distribution to the problem of authenticated distribution of the *CA*'s public key, at the expense of a trusted centre (the *CA*), see reference ISO/IEC 9594-8 and ISO/IEC 11770-1:1996, Annex D.

12.2.1 Public key transport mechanism 3

This mechanism transfers a public key from entity *A* to entity *B* in an authenticated way. It is based on the assumption that a valid public key certificate $Cert_A$ of entity *A*'s public key information PKI_A has been issued by some certification authority, and that entity *B* has access to an authenticated copy of the public verification transformation V_{CA} of that certification authority *CA* which has issued the public key certificate.

See Figure 17 for a diagram of public key transport mechanism 3.

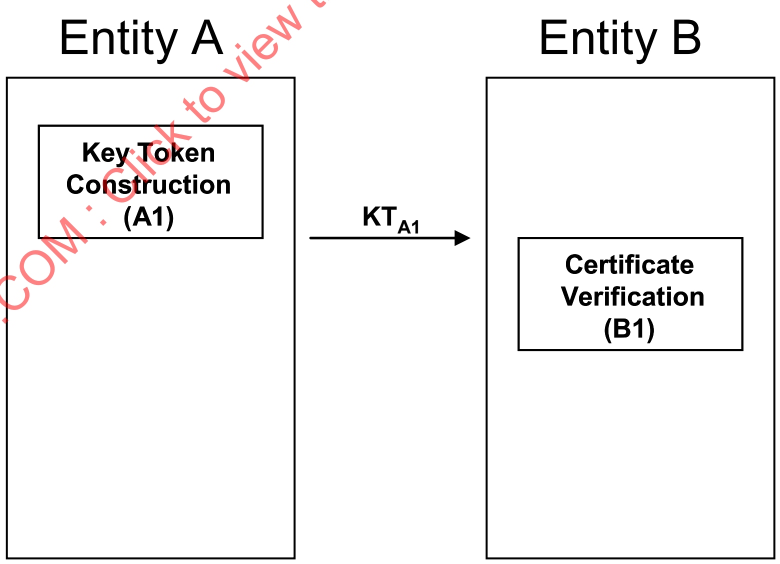


Figure 17: Public Key Transport Mechanism 3

Key Token Construction (A1) Entity *A* constructs the key token KT_{A1} containing the public key certificate of entity *A* and sends it to entity *B*

$$KT_{A1} = Cert_A \,||\, Text.$$

Certificate Verification (B1) Upon reception of the public key certificate, entity *B* uses the public verification transformation V_{CA} of the certification authority to verify the authenticity of the public key information and to check the validity of entity *A*'s public key.

If entity *B* wants to make sure that entity *A*'s public key certificate has not been revoked recently, then entity *B* should consult a trusted third party (such as the *CA*) via some authenticated channel.

NOTES - This Public Key Transport Mechanism has the following properties:

1) Number of passes: 1. But there may have been a request from entity *B* to entity *A* for the transfer of the public key certificate. This additional pass is optional and not shown here. Entity *A*'s public key certificate could also be distributed by a directory, in which case this public key transport mechanism would be executed between the directory and entity *B*.

2) Entity authentication: entity authentication is not provided by this mechanism.

3) Key confirmation: receiving a public key certificate provides confirmation that the public key has been certified by the *CA*.

4) The public verification key v_{CA} of the *CA* shall be made available to entity *B* in an authenticated way. This can be done using the mechanisms described in Clause 12.

Annex A
(informative)

Properties of key establishment mechanisms

The following tables summarize the major properties of the key establishment/transport mechanisms specified in this part of ISO/IEC 11770.

The following notation is used:

A	mechanism provides the property with respect to entity *A*.
B	mechanism provides the property with respect to entity *B*.
A, B	the mechanism provides the property with respect to both entities, *A* and *B*.
No	the mechanism does not provide the property.
Opt	the mechanism can provide the property as an option, using additional means.
(A)	the mechanism can optionally provide the property with respect to entity *A*, using additional means.
(B)	the mechanism can optionally provide the property with respect to entity *B*, using additional means.
MFS	the mechanism provides mutual forward secrecy.

Public key operations: the number of computations of asymmetric transformation, e.g., "2,1" means that entity *A* needs two computations of the function *F* and entity *B* needs one computation of the function *F* in Key Agreement Mechanism 2.

Another important property that can be derived from key freshness is replay attack prevention. Replay attacks are generally not possible where key freshness is guaranteed for both entities.

The property of implicit key authentication has direction by its definition. When the table for implicit key authentication has an "A", this means that entity *A* is assured that entity *B* is the only other entity that can possibly be in possession of the correct key. When the table for implicit key authentication has an "A, B", this means that entities *A* and *B* are assured that only the other entity can possibly be in possession of the correct key.

Properties of Key Agreement Mechanisms:

Mechanism	1	2	3	4	5	6	7	8	9	10	11
Number of passes	0	1	1	2	2	2	3	1	2	3	4
Implicit key authentication	A, B	B	A, B	No	A, B	A, B	A, B	A, B	A, B	A, B	A, B
Key confirmation	No	No	B	No	Opt	Opt	A, B	No	No	A, B	A, B
Entity authentication	No	No	A	No	No	B	A, B	No	No	A, B	B
Public key operations	1, 1	2, 1	2, 2	2, 2	3, 3	2, 2	4, 4	2, 1	2, 2	2, 2	4, 3
Forward secrecy	No	A	A	MFS	A,B	B	MFS	A	MFS	MFS	MFS
Key freshness	No	A	A	A, B	A, B	A, B	A, B	A	A, B	A, B	A, B

Properties of Key Transport Mechanisms:

Mechanism	1	2	3	4	5	6
Number of passes	1	1	1	2	3	3
Implicit key authentication	B	B	B	A	A, B	A, B
Key confirmation	No	B	B	A	(A), B	A, B
Key control	A	A	A	B	A resp. B	A, B
Entity authentication	No	A	A	B	A, B	A, B
Public key operations	1, 1	2, 2	2, 2	2, 2	4, 4	2, 2
Forward secrecy	A	A	A	B	No	No
Key freshness	A	A	A	A	A, B	A, B

Annex B
(informative)

Examples of key derivation functions

B.1 The IEEE P1363 key derivation function

This clause describes the key derivation function that is given in the IEEE P1363 standard [11].

Preconditions As a precondition of the use of this key derivation function, users must agree on a common hash function. Users who use different hash functions will obtain different results. For the purposes of this part of ISO/IEC 11770, the hash function shall be one described in ISO/IEC 10118. The shared key that is produced will have length equal to the length of the output of the hash function.

Input The inputs to this key derivation function are

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

NOTE The mechanisms in Clauses 10 and 11 derive shared keys K_{AB} either as points on the elliptic curve or as the concatenation of two points on an elliptic curve. In the first situation, in order to obtain a shared secret integer *z* for input into the key derivation function, the function π should be applied to the point. In the second situation, the function π should be applied to both points to obtain two integers z_1 and z_2. The two integers should then be converted to octet strings and concatenated (or combined using any prefix-free encoding method), as were the points, to obtain the appropriate octet string.

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

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

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

Otherwise compute the value $K = hash(z \,||\, parameters)$.

Output Output *K* as the shared secret.

B.2 The ANSI X9.42 key derivation function

This clause describes a key derivation function based on the key derivation function that is given in the ANSI X9.42 standard [4].

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

Input The input to the key derivation function is:

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

NOTE The mechanisms in Clauses 10 and 11 derive shared keys K_{AB} either as points on the elliptic curve or as the concatenation of two points on an elliptic curve. In the first situation, in order to obtain a shared secret value *ZZ* for input into the key derivation function, the function π should be applied to the point and the resulting integer converted to a bit string. In the second situation, the function π should be applied to both points to obtain two integers z_1 and z_2. The two integers should then be converted to bit strings and concatenated (or combined using any prefix-free encoding method), as were the points, to obtain the appropriate bit string.

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

3) OtherInfo: A bit string, specified in ASN.1 DER encoding, consisting of the following information as specified in Annex B.2.1.

3.1. Key specification information consisting of:

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

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

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

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

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

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

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

Actions The key derivation function is computed as follows:

1) Let d = ⌈keydatalen / hashlen⌉.

2) Initialise Counter = 00000001_{16}.

3) For i = 1 to d,

3.1. Compute h_i = *hash*(ZZ || OtherInfo) where h_i denotes the hash value computed using the appropriate hash function, and OtherInfo = AlgorithmID || Counter [|| EntityAInfo || EntityBInfo || SuppPrivInfo || SuppPubInfo].

3.2. Increment Counter.

3.3. Increment i.

4) Compute K = leftmost keydatalen bits of h_1 || h_2 || … || h_d.

5) Output K.

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

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

B.2.1 ASN.1 syntax

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

The other information includes the initiator’s information entityAInfo, and the responder’s information entityBInfo, suppPubInfo, and suppPrivInfo.

```
OtherInfo ::= SEQUENCE {

    keyInfo   KeySpecificInfo,

    entityAInfo    [0] OCTET STRING  OPTIONAL,

    entityBInfo    [1] OCTET STRING  OPTIONAL,

    suppPubInfo [2] OCTET STRING  OPTIONAL,

    suppPrivInfo  [3] OCTET STRING  OPTIONAL

}

KeySpecificInfo ::= SEQUENCE {

    algorithm      OBJECT IDENTIFIER,

    counter   Counter

}

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

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

The contents of suppPubInfo and suppPrivInfo are defined by the key management protocol. The definition, syntax, and encoding rules of the suppPubInfo and suppPrivInfo fields are the responsibility of the key management protocol and are beyond the scope of this part of ISO/IEC 11770.

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

B.3 The ANSI X9.63 key derivation function

This clause describes a key derivation function based on the key derivation function that is given in the ANSI X9.63 standard [5].

Prerequisites The prerequisite for the operation of the key derivation function is that a hash function, hash, specified in ISO/IEC 10118 has been chosen. Let hashlen denote the length of the output of the hash function chosen, and let maxhashlen denote the maximum length of the input to the hash function.

Input The input to the key derivation function is:

1) A bit string Z which is the shared secret.

NOTE The mechanisms in Clauses 10 and 11 derive shared keys K_{AB} either as points on the elliptic curve or as the concatenation of two points on an elliptic curve. In the first situation, in order to obtain a shared secret *Z* for input into the key derivation function, the function π should be applied to the point and the resulting integer converted to a bit string. In the second situation, the function π should be applied to both points to obtain two integers z_1 and z_2. The two integers should then be converted to bit strings and concatenated (or combined using any prefix-free encoding method), as were the points, to obtain the appropriate bit string.

2) An integer keydatalen which is the length in bits of the keying data to be generated. keydatalen shall be less than hashlen×(2^{32}–1).

3) (Optional) A bit string SharedInfo which consists of some data shared by the two entities intended to share the secret Z.

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

Actions The key derivation function is computed as follows:

1) Initiate a 32-bit, big-endian bit string counter as 00000001_{16}.

2) For i = 1 to j = ⌈keydatalen / hashlen⌉, do the following:

2.1 Compute Hash_i = H(Z || counter [|| SharedInfo]).

2.2 Increment counter.

2.3 Increment i.

3) Let HHash_j denote Hash_j if keydatalen/hashlen is an integer, and let it denote the (keydatalen - (hashlen×(j-1))) leftmost bits of Hash_j otherwise.

4) Set KeyData = Hash_1 || Hash_2 || ... || Hash_{j-1} || HHash_j.

Output The bit string KeyData of length keydatalen bits.

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

B.4 The NIST SP 800-56A concatenation key derivation function

This clause describes a key derivation function based on the key derivation function that is given in the NIST Special Publication 800-56A [20].

Function call: kdf (Z, OtherInput),

where OtherInput is keydatalen and OtherInfo.

Fixed Values (implementation dependent):

1) hashlen: an integer that indicates the length (in bits) of the output of the hash function used to derive blocks of secret keying material.

2) max_hash_inputlen: an integer that indicates the maximum length (in bits) of the bit string(s) input to the hash function.

Auxiliary Function:

1) *H*: an approved hash function specified in ISO/IEC 10118.

Input:

1) *Z*: a byte string that is the shared secret.

2) keydatalen: An integer that indicates the length (in bits) of the secret keying material to be generated; keydatalen shall be less than or equal to hashlen × (2^{32} –1).

3) OtherInfo: A bit string equal to the following concatenation:

AlgorithmID || EntityAInfo || EntityBInfo [|| SuppPubInfo] [|| SuppPrivInfo]

where the subfields are defined as follows:

3.1) AlgorithmID: A bit string that indicates how the derived keying material will be parsed and for which algorithm(s) the derived secret keying material will be used. For example, AlgorithmID might indicate that bits 1-80 are to be used as an 80-bit HMAC key and that bits 81-208 are to be used as a 128-bit AES key.

3.2) EntityAInfo: A bit string containing public information that is required by the application using this kdf to be contributed by entity *A* to the key derivation process. At a minimum, EntityAInfo shall include ID_A, the identifier of entity *A*. See the notes below.

3.3) EntityBInfo: A bit string containing public information that is required by the application using this kdf to be contributed by entity *B* to the key derivation process. At a minimum, EntityBInfo shall include ID_B, the identifier of entity *B*. See the notes below.

3.4) (Optional) SuppPubInfo: A bit string containing additional, mutually-known public information.

3.5) (Optional) SuppPrivInfo: A bit string containing additional, mutually-known private information (for example, a shared secret symmetric key that has been communicated through a separate channel).

Each of the three subfields AlgorithmID, EntityAInfo, and EntityBInfo shall be the concatenation of an application-specific, fixed-length sequence of substrings of information. Each substring representing a separate unit of information shall have one of these two formats: Either it is a fixed-length bit string, or it has the form Datalen || Data, where Data is a variable-length string of zero or more bytes, and Datalen is a fixed-length, big-endian counter that indicates the length (in bytes) of Data. (In this variable-length format, a null string of data shall be represented by using Datalen to indicate that Data has length zero.) An application using this kdf shall specify the ordering and number of the separate information substrings used in each of the subfields AlgorithmID, EntityAInfo, and EntityBInfo, and shall also specify which of the two formats (fixed-length or variable-length) is used for each substring. The application shall specify the lengths for all fixed-length quantities, including the Datalen counters.

The subfields SuppPrivInfo and SuppPubInfo (when allowed by the application) shall be formed by the concatenation of an application-specific, fixed-length sequence of substrings of additional information that may be used in key derivation upon mutual agreement of entities *A* and *B*. Each substring representing a separate unit of information shall be of the form Datalen || Data, where Data is a variable-length string of zero or more (eight-bit) bytes and Datalen is a fixed-length, big-endian counter that indicates the length (in bytes) of Data. The information substrings that entities *A* and *B* choose not to contribute are set equal to Null, and are represented in this variable-length format by setting

Datalen equal to zero. If an application allows the use of the OtherInfo subfield SuppPrivInfo and/or the subfield SuppPubInfo, then the application shall specify the ordering and the number of additional information substrings that may be used in the allowed subfield(s) and shall specify the fixed-length of the Datalen counters.

Process:

1) reps = $\lceil$ keydatalen / hashlen $\rceil$.

2) If reps > (2^{32} −1), then ABORT: output an error indicator and stop.

3) Initialize a 32-bit, big-endian bit string counter as 00000001_{16}.

4) If counter || Z || OtherInfo is more than max_hash_inputlen bits long, then ABORT: output an error indicator and stop.

5) For i = 1 to reps by 1, do the following:

   5.1) Compute Hash_i = H(counter || Z || OtherInfo).

   5.2) Increment counter (modulo 2^{32}), treating it as an unsigned 32-bit integer.

6) Let Hhash be set to Hash_{reps} if (keydatalen / hashlen) is an integer; otherwise, let Hhash be set to the (keydatalen mod hashlen) leftmost bits of Hash_{reps}.

7) Set DerivedKeyingMaterial = Hash_1 || Hash_2 || … || Hash_{reps-1} || Hhash.

Output: The bit string DerivedKeyingMaterial of length keydatalen bits (or an error indicator). Any scheme attempting to call this key derivation function with keydatalen greater than or equal to hashlen × (2^{32} −1) shall output an error indicator and stop without outputting DerivedKeyingMaterial. Any call to the key derivation function involving an attempt to hash a bit string that is greater than max_hash_inputlen bits long shall cause the kdf to output an error indicator and stop without outputting DerivedKeyingMaterial.

NOTES

1) ID_A and ID_B shall be represented in OtherInfo as separate units of information, using either the fixed-length format or the variable-length format described above – according to the requirements of the application using this kdf.

2) Entity *A* shall be the initiator, and entity *B* shall be the responder, as assigned by the protocol employing the key agreement scheme used to determine the shared secret *Z*.

B.5 The NIST SP 800-56A ASN.1 key derivation function

This clause describes a key derivation function based on the key derivation function that is given in the NIST Special Publication 800-56A [20].

Function call: *kdf* (Z, OtherInput)

where OtherInput is keydatalen and OtherInfo.

Fixed Values (implementation dependent):

1) hashlen: an integer that indicates the length (in bits) of the output of the hash function used to derive blocks of secret keying material.

2) max_hash_inputlen: an integer that indicates the maximum length (in bits) of the bit string(s) input to the hash function.

Auxiliary Function:

1) *H*: an approved hash function specified in ISO/IEC 10118.

Input:

1) *Z*: a byte string that is the shared secret.

2) keydatalen: An integer that indicates the length (in bits) of the secret keying material to be generated; keydatalen shall be less than or equal to hashlen × (2^{32} –1).

3) OtherInfo: A bit string specified in ASN.1 DER encoding, which consists of the following information:

 3.1) AlgorithmID: A bit string that indicates how the derived keying material will be parsed and for which algorithm(s) the derived secret keying material will be used. For example, AlgorithmID might indicate that bits 1-80 are to be used as an 80-bit HMAC key and that bits 81-208 are to be used as a 128-bit AES key.

 3.2) EntityAInfo: A bit string containing public information that is required by the application using this kdf to be contributed by entity *A* to the key derivation process. At a minimum, EntityAInfo shall include ID_A, the identifier of entity *A*. See the notes below.

 3.3) EntityBInfo: A bit string containing public information that is required by the application using this kdf to be contributed by entity *B* to the key derivation process. At a minimum, EntityBInfo shall include ID_B, the identifier of entity *B*. See the notes below.

 3.4) (Optional) SuppPubInfo: A bit string containing additional, mutually-known public information.

 3.5) (Optional) SuppPrivInfo: A bit string containing additional, mutually-known private information (for example, a shared secret symmetric key that has been communicated through a separate channel).

Process:

1) reps = ⌈ keydatalen / hashlen ⌉.

2) If reps > (2^{32} −1), then ABORT: output an error indicator and stop.

3) Initialize a 32-bit, big-endian bit string counter as 00000001_{16}.

4) If counter || Z || OtherInfo is more than max_hash_inputlen bits long, then ABORT: output an error indicator and stop.

5) For i = 1 to reps by 1, do the following:

 5.1) Compute $Hash_i$ = H(counter || Z || OtherInfo).

 5.2) Increment counter (modulo 2^{32}), treating it as an unsigned 32-bit integer.

6) Let Hhash be set to $Hash_{reps}$ if (keydatalen / hashlen) is an integer; otherwise, let Hhash be set to the (keydatalen mod hashlen) leftmost bits of $Hash_{reps}$.

7) Set DerivedKeyingMaterial = $Hash_1$ || $Hash_2$ || … || $Hash_{reps-1}$ || Hhash.

Output: The DerivedKeyingMaterial as a bit string of length keydatalen bits (or an error indicator). The ASN.1 kdf produces secret keying material that is at most hashlen × (2^{32}–1) bits in length. Any call to this key derivation function using a keydatalen value that is greater than hashlen × (2^{32}–1) shall cause the kdf to output an error indicator and stop without outputting DerivedKeyingMaterial. Any call to the key derivation function involving an attempt to hash a bit string that is greater than max_hash_inputlen bits long shall cause the kdf to output an error indicator and stop without outputting DerivedKeyingMaterial.

NOTES

1) ID_A and ID_B shall be represented in OtherInfo as separate units of information.

2) Entity *A* shall be the initiator, and entity *B* shall be the responder, as assigned by the protocol employing the key agreement scheme used to determine the shared secret *Z*.

Annex C
(informative)

Examples of key establishment mechanisms

This informative annex gives examples of some of the key establishment mechanisms described in this part of ISO/IEC 11770.

We first specify a widely used example of a function *F*, and accompanying sets *S* and *H*, which is conjectured to satisfy the five properties listed in Clause 10, given that certain parameters are chosen appropriately.

Let *p* be a prime number, *S* be the set of elements of the Galois field with *p* elements, F_p, and let *H* = {2, ... , p - 2}. Let *g* be a primitive element of F_p. Then set

$$F(h,g) = g^h \bmod p.$$

F is commutative with respect to *h*

$$(g^{h_B})^{h_A} = (g^{h_A})^{h_B} = g^{h_A h_B} \bmod p.$$

The prime *p* must be large enough so that *F(·,g)* can be conjectured to be a one-way function. Let each entity *X* have a private key h_X in *H*, which is only known by entity *X*, and a public key $p_X = g^{hx} \bmod p$ known by all other entities.

> NOTE - On the selection of parameters.
>
> For discrete logarithm modulo a prime: The size of the prime should be chosen such that computing discrete logarithms in the corresponding cyclic group is computationally infeasible. Some other conditions on the prime number may be imposed in order to make discrete logarithms infeasible.
>
> It is recommended to either choose p to be a strong prime such that *p - 1* has a large prime factor or to choose *g* to be a generator of a group of large prime order *q*.
>
> For discrete logarithm modulo a composite: The modulus should be chosen as the product of two distinct odd primes that should be kept secret. The size of the modulus should be chosen such that factoring the modulus is computationally infeasible. Some additional conditions on the choice of the primes may be imposed in order to make factoring the modulus computationally infeasible.

C.1 Non-interactive Diffie-Hellman key agreement

This [8] is an example of Key Agreement Mechanism 1.

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 key as

$$K_{AB} = p_B^{\ h_A} \bmod p.$$

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 key as

$$K_{AB} = p_A^{\ h_B} \bmod p.$$

C.2 Identity-based mechanism

This [10] is an example of Key agreement Mechanism 1, which is identity-based in the following sense:

- the public key of an entity can be retrieved from some combination of its identity and its certificate;

- the authenticity of the certificate is not directly verified, but the correct public key can only be recovered from an authentic certificate.

Let *(n,y)* be the public verification key of a certification authority, in the digital signature scheme giving message recovery specified in ISO/IEC 9796. Therefore *n* is the product of two large prime numbers *p* and *q*, kept secret by the certification authority, and *y* is co-prime with *lcm(p-1, q-1)*.

Let *O* be an integer of large order modulo *n* and $g = O^y \bmod n$.

Let I_X be the result of adding redundancy (as specified in ISO/IEC 9796) to a public information on entity *X* which contains at least the distinguished identifier of entity *X* and possibly a serial number, a validity period, a time stamp and other data elements. Then entity *X*'s key management pair is (h_X, p_X) where h_X is an integer less than *n* and

$$p_X = g^{h_X} \bmod n.$$

Its certificate is computed by the certification authority as

$$\mathrm{Cert}_X = s_X O^{h_X} \bmod n,$$

where s_X is the integer such that

$$s_X{}^Y I_X = 1 \bmod n.$$

Key Construction (A1) Entity *A* computes the public key of entity *B* as

$$p_B = \mathrm{Cert}_B{}^Y \cdot I_B \bmod n$$

and computes the shared secret key as

$$K_{AB} = p_B{}^{h_A} = g^{h_A h_B} \bmod n.$$

Key Construction (B1) Entity *B* computes the public key of entity *A* as

$$p_A = \mathrm{Cert}_A{}^Y \cdot I_A \bmod n$$

and computes the shared secret key as

$$K_{AB} = p_A{}^{h_B} = g^{h_A h_B} \bmod n.$$

NOTE A one-pass and a two-pass identity-based mechanisms using the same set-up are described in the references [10], [22] and [24] in the Bibliography.

C.3 ElGamal key agreement

This [9] is an example of Key Agreement Mechanism 2.

One shall check that *p* to be a strong prime such that *p - 1* has a large prime factor and that the exponentials are not of the form *0*, *+1*, *-1* mod *p*.

Key Token Construction (A1) Entity *A* randomly and secretly generates *r* in {*2, ... , p-2* }, computes g^r mod *p* and constructs the key token

$$KT_{A1} = g^r \bmod p$$

and sends it to entity *B*.

Key Construction (A2) Entity *A* computes the shared key

$$K_{AB} = (p_B)^r \bmod p = g^{h_B r} \bmod p.$$

Key Construction (B1) Entity *B* computes the shared key

$$K_{AB} = (g^r)^{h_B} \bmod p = g^{h_B r} \bmod p.$$

C.4 Nyberg-Rueppel key agreement

This [21] is an example of Key Agreement Mechanism 3. The signature system and the key agreement system are chosen in such a way that the signature system is determined by the keys (h_X , p_X).

Let *q* be a large prime divisor of *p-1*, *g* an element of F_p of order *q*, and set *H* = {*2, ... , q-2*}. Then entity *X*'s asymmetric key pair used for signatures and key agreements is (h_X, p_X), where h_X is an element of *H* and

$$p_X = g^{h_X} \bmod p.$$

To prevent replay of old key tokens this example makes use of a time-stamp or a serial number, *TVP*, and of a cryptographic hash function hash, which maps strings of bits of arbitrary length to random integers in a large subset of {*2, ... , p-2*}, for example, in *H*.

Key Construction (A1.1) Entity *A* randomly and secretly generates r in *H* and computes

$$e = g^r \bmod p.$$

Further entity *A* computes the shared secret key as

$$K_{AB} = p_B{}^r \bmod p.$$

Using the shared secret key K_{AB}, entity *A* computes a *MAC* on the sender's distinguished identifier for entity *A* and a sequence number or time-stamp *TVP*

$$e' = e\,\text{hash}(K_{AB} \,||\, A \,||\, TVP) \bmod p.$$

Key Token Signature (A1.2) Entity *A* computes the signature

$$y = r - h_A\, e' \bmod q.$$

Entity *A* forms the key token

$$KT_{A1} = A \,||\, e \,||\, TVP \,||\, y$$

and sends it to entity *B*.

Key Construction (B1.1) Entity *B* computes the shared secret key, using its private key agreement key h_B,

$$K_{AB} = e^{h_B} \bmod p.$$

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

$$e' = e \, \mathrm{hash}(K_{AB} \,||\, A \,||\, TVP) \bmod p.$$

Signature Verification (B1.2) Entity *B* checks the validity of *TVP* and verifies, using the sender's public key p_A, the equality

$$e = g^y \, p_A^{e'} \bmod p.$$

C.5 Diffie-Hellman key agreement

This [8] is an example of Key Agreement Mechanism 4.

One shall check that p to be a strong prime such that *p - 1* has a large prime factor and that the exponentials are not of the form *0, +1, -1* mod *p*.

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in {*2, ... , p-2* }, computes g^{r_A} mod *p*, constructs the key token

$$KT_{A1} = g^{r_A} \bmod p$$

and sends it to entity *B*.

Key Token Construction (B1) Entity *B* randomly and secretly generates r_B in {*2, ... , p-2* }, computes g^{r_B} mod *p*, constructs the key token

$$KT_{B1} = g^{r_B} \bmod p$$

and sends it to entity *A*.

Key Construction (A2) Entity *A* computes the shared key

$$K_{AB} = (g^{r_B})^{r_A} = g^{r_A r_B} \bmod p.$$

Key Construction (B2) Entity *B* computes the shared key

$$K_{AB} = (g^{r_A})^{r_B} = g^{r_A r_B} \bmod p.$$

C.6 Matsumoto-Takashima-Imai A(0) key agreement

This [1] is an example of Key Agreement Mechanism 5.

One recommended method is to use a safe prime *p* and to check that the exponentials are not of the form 0, +1, -1 mod p.

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in {2, ... , p-2}, computes the key token

$$KT_{A1} = g^{r_A} \bmod p$$

and sends it to entity *B*.

Key Token Construction (B1) Entity *B* randomly and secretly generates r_B in {2, ... , p-2}, computes the key token

$$KT_{B1} = g^{r_B} \bmod p$$

and sends it to entity *A*.

Key Construction (B2) Entity *B* computes the shared key as

$$K_{AB} = w(KT_{A1}{}^{h_B}, p_A{}^{r_B}) = KT_{A1}{}^{h_B}\ p_A{}^{r_B} \bmod p.$$

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

$$K_{AB} = w(p_B{}^{r_A}, KT_{B1}{}^{h_A}) = KT_{A1}{}^{h_B}\ p_A{}^{r_B} \bmod p.$$

C.7 Beller-Yacobi protocol

This part of the Annex gives a description of the original Beller-Yacobi protocol [6], which has been used to derive Key Agreement Mechanism 6.

> NOTE This mechanism is not completely compatible with the Mechanism 6 as it was optimized for specific situations. Specifically it uses ElGamal signature scheme and makes use of an additional symmetric encryption algorithm to transfer entity *B*'s signature verification key and its certificate to entity *A* in a confidential way, thus assuring anonymity.

Let enc: $K : M \rightarrow C$ be a conventional encryption function, such as the algorithms found in ISO/IEC 18033-3, where K = key space, M = message space, and C = cryptogram space.

Let S_X denote the ElGamal signature operation of entity *X*. The process described below emphasizes the distinction between off-line and on-line operations required in ElGamal family of signature schemes.

We use P_X and C_X to denote entity *X*'s public key and certificate, respectively. The public encryption operation of entity X (which uses P_X) is denoted E_X (modular squaring in the case of Rabin).

Off-line computation: entity *B* picks a random number r_B and computes

$$u = g^{r_B} \bmod p.$$

Key Token Construction (A1) Entity *A* picks a random number r_A and computes

$$KT_{A1} = (r_A \,||\, A \,||\, C_A)$$

and sends it to entity *B*.

Key Token Processing (B1) Entity *B* produces the signature

$$BS = (u, v) = S_B(r_A \,||\, A),$$

where *u* and *v* is the ElGamal signature. Then entity *B* picks a random x_B and creates

$$KT_{B1} = E_A(BS) \,||\, enc(u, (B \,||\, P_B \,||\, C_B \,||\, x_B))$$

and sends it to entity *A*.

Key Construction (B2) The shared secret key consists of part of entity *B*'s signature, *u*.

Entity Authentication and Key Construction (A2) Entity *A* deciphers the key token $E_A(BS)$ to find the session key *u*, then deciphers the conventional encryption

$$enc(u, (B \,||\, P_B \,||\, C_B \,||\, x_B))$$

using session key *u* to find the identifier, public key, and certificate of the alleged entity *B*. Entity *A* verifies certificate C_B, and if positive it then uses the verification function, V_B to verify entity *B*'s signature *BS*. If positive it then accepts *u* as a shared secret key.

Annex D
(informative)

Examples of elliptic curve based key establishment mechanisms

This informative annex gives examples of some of the elliptic curve key establishment mechanisms described in this part of ISO/IEC 11770. See ISO/IEC 15946-1 for information about elliptic curves and their parameters for cryptographic applications.

We first specify a widely used example of a function *F* to satisfy the five properties listed in Clause 10, given that certain parameters are chosen appropriately.

Let *E* be an elliptic curve defined over a finite field *F(q)*. Given an integer *d* and a point $G \in E(F(q))$ where *G* is the base point, then the function *F* is

$$\mathrm{F(d,G) = d\,G.}$$

F has the property that

$$\mathrm{d_1(d_2\ G) = d_2(d_1\ G) = d_1 d_2\ G.}$$

E(F(q)) must be large enough so that $F(\cdot,G)$ can be conjectured to be a one-way function. Let each entity *X* have a private key h_X in *E(F(q))*, which is only known by entity *X*, and a public key $p_X = h_X\ G$ known by all other entities.

D.1 Common information

For all key agreement mechanisms, prior to the process of agreeing upon a shared secret, the following common information must be established between the parties and optionally validated (See ISO/IEC 15946-1 for a description of parameter validation):

— the elliptic curve parameters with which the key pairs shall be associated, which shall be the same for both parties key pairs. This includes *p*, p^m, 2^m, or 3^m, a description of *F(p)*, $F(p^m)$, $F(2^m)$, or $F(3^m)$ and an indication of the basis used, *E*, *n* and *G*.

Named curve identifiers such as those specified in X9.62, provide a simple means of identifying elliptic curve domain parameters and can be used to specify groups of common information values.

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

— a key derivation function, *kdf*;

— any parameters to the key derivation function, and

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

D.2 Non-interactive key agreement of Diffie-Hellman type

This [8] is an example of key agreement 1. This key agreement mechanism non-interactively establishes a shared secret between two entities *A* and *B*.

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

— for each entity *X*, a private key-agreement key d_X and a public key-agreement key P_X, which is an elliptic curve point satisfying $P_X = d_X G$. See ISO/IEC 15946-1 for a description of how to generate this key pair.

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

Each entity should independently verify that the other entity‘s public key is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key construction (A1) Entity *A* computes, using its own private key-agreement key d_A and entity *B*’s public key-agreement key P_B, the shared key as

$$K_{AB} = (d_A \cdot l)(j \cdot P_B).$$

Key construction (B1) Entity *B* computes, using its own private key-agreement key d_B and entity *A*’s public key-agreement key P_A, the shared key as

$$K_{AB} = (d_B \cdot l)(j \cdot P_A).$$

NOTES - This key agreement mechanism has the following properties:

1) Number of passes: *0*.

2) The mechanism provides mutual implicit key authentication.

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

D.3 Key agreement of ElGamal type

This [9] is an example of key agreement mechanism 2. This key agreement mechanism establishes a shared secret between two entities *A* and *B* in one pass.

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

— for entity *B*, a private key-agreement key d_B and a public key-agreement key P_B, which is an elliptic curve point satisfying $P_B = d_B G$. See ISO/IEC 15946-1 for a description of how to generate this key pair.

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

Entity *A* should verify that entity *B*’s public key is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key token construction (A1.1) Entity *A* randomly and secretly generates *r* in the range {*2,...,n-2*}, computes *rG*, constructs the key token,

$$KT_{A1} = rG$$

and sends it to entity *B*.

Key construction (A1.2) Entity *A* computes the shared key

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

Key construction (B1) Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Using its own private key, entity *B* computes the shared key from KT_{A1} as follows:

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

NOTES - This key agreement mechanism has the following properties:

1) Number of passes: 1.

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

3) The mechanism provides forward secrecy with respect to entity A.

D.4 Key agreement following Nyberg-Rueppel

This [21] is an example of Key Agreement Mechanism 3. The protocol is not a 1-1-transcript of protocol C.4; but follows the essential ideas of C.4.

The signature system and the key agreement system are chosen in such a way that the signature system is determined by the keys (h_X, P_X).

To prevent the replay of old key tokens this example makes use of a timestamp or a serial number *TVP*, and of a cryptographic hash function hash, which maps strings of bits of arbitrary length to random integers into *H*, for example.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key Construction (A1.1) Entity *A* randomly and secretly generates *r* in *H* and computes

$$R = r\,G.$$

Further entity *A* computes the shared secret key as

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

Using the shared secret key K_{AB}, entity *A* computes a *MAC* on the point *R*, the sender's distinguished identifier for entity *A* and a sequence number or timestamp *TVP*:

$$e = \text{hash}(R \,||\, K_{AB} \,||\, A \,||\, TVP).$$

Key Token Signature (A1.2) Entity *A* computes the signature

$$y = (r - h_A\ e) \bmod q,$$

forms the key token

$$KT_{A1} = (R \,||\, A \,||\, TVP \,||\, y)$$

and sends it to entity *B*.

Key Construction (B1.1) Entity *B* computes the shared secret key, using its private key agreement key h_B,

$$K_{AB} = (h_B \cdot l)(j \cdot R).$$

Using the shared secret key K_{AB} entity *B* computes the *MAC* on the sender's distinguished identifier for entity *A* and the *TVP* and computes

$$e = \mathrm{hash}(R \,||\, K_{AB} \,||\, A \,||\, TVP).$$

Signature Verification (B1.2) Entity *B* checks the validity of *TVP* and verifies, using the sender's public key P_A, the equality

$$R = y\,G + e\,P_A.$$

This key agreement has the following properties:

1) Number of passes: *1*.

2) The mechanism provides explicit key authentication from entity *A* to entity *B* and implicit key authentication from entity *B* to entity *A*.

D.5 Key agreement of Matsumoto-Takashima-Imai type A(0)

This [1] is an example of Key Agreement Mechanism 5. The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

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

$$KT_{A1} = (r_A \cdot l)(j \cdot G),$$

and sends it to entity *B*.

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

$$KT_{B1} = (r_B \cdot l)(j \cdot G)$$

and sends it to entity *A*.

Key Construction (B2) Entity *B* computes the shared key as

$$K_{AB} = w(h_B\ KT_{A1}, r_B\ P_A\ )$$

where *w* is a one-way function.

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

$$K_{AB} = w(h_A\ KT_{B1}, r_A\ P_B).$$

This key agreement has the following properties:

1) Number of passes: *2*

2) The mechanism provides mutual implicit key authentication.

D.6 Key agreement of Diffie-Hellman type

This [8] is an example of key agreement mechanism 4. This key agreement mechanism establishes a shared secret between entities *A* and *B* in two passes.

This key agreement mechanism does not require any initial information other than the common information to be set up. The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in the range {*2,...,n-2*}, computes r_AG, constructs the key token,

$$KT_{A1} = r_A G$$

and sends it to entity *B*.

Key Token Construction (B1) Entity *B* randomly and secretly generates r_B in the range {*2,...,n-2*}, computes r_BG, constructs the key token,

$$KT_{B1} = r_B G$$

and sends it to entity *A*.

Key Construction (A2) Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *A* computes the shared key

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

Key Construction (B2) Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *B* computes the shared key

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

NOTES - This key agreement mechanism has the following properties:

1) Number of passes: *2*.

2) The mechanism provides mutual forward secrecy.

D.7 Key agreement of Diffie-Hellman type with 2 key pairs

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

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

— for each entity X, a private key-agreement key d_X and a public key-agreement key P_X, which is an elliptic curve point satisfying $P_X = d_XG$. See ISO/IEC 15946-1 for a description of how to generate this key pair.

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

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key token construction (A1) Entity *A* randomly and secretly generates r_A in the range $\{2,...,n-2\}$, computes r_AG, constructs the key token,

$$KT_{A1} = r_A G$$

and sends it to entity *B*.

Key token construction (B1) Entity *B* randomly and secretly generates r_B in the range $\{2,...,n-2\}$, computes r_BG, constructs the key token,

$$KT_{B1} = r_B G$$

and sends it to entity *A*.

Key construction (A2): Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *A* computes the shared key

$$K_{AB} = (d_A \cdot l)(j \cdot KT_{B1}) \,||\, (r_A \cdot l)(j \cdot P_B).$$

Key construction (B2) Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *B* computes the shared secret

$$K_{AB} = (r_B \cdot l)(j \cdot P_A) \,||\, (d_B \cdot l)(j \cdot KT_{A1}).$$

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

NOTES - This key agreement mechanism has the following properties:

1) Number of passes: *2*.

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

3) The mechanism provides mutual implicit key authentication.

D.8 Key agreement of Diffie-Hellman type with 2 signatures and key confirmation

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

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

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

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

— any parameters to be used in the signature transformations.

— a *MAC* function.

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

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key token construction (A1) Entity *A* randomly and secretly generates r_A in the range $\{2,...,n-2\}$, computes $r_A G$, constructs the key token

$$KT_{A1} = r_A G$$

and sends it to entity *B*.

Key token construction (B1) Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *B* randomly and secretly generates r_B in the range $\{2,...,n-2\}$, computes $r_B G$, computes the shared secret as

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

constructs the signed key token,

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

where

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

and sends it to entity *A*.

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

Key token processing (A2) Entity *A* verifies *B*'s signature on the key token KT_{B1} using entity *B*'s public verification key. If a signature scheme with message recovery is used, this includes recovering the data block DB_1 from the signature and verifying that entity *A*'s distinguishing identifier and the value $r_A G$ are contained in it. If a signature scheme with appendix is used, this includes reconstructing the data block DB_1 using the value in KT_{A1}, entity *A*'s distinguishing identifier and the received value $r_B G$ and verifying the signature on that data block.

Entity *A* should verify that the value $r_B G$ obtained from KT_{B1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. If the checks are successful, entity *A* computes the shared key

$$K_{AB} = (r_A \cdot l)(j \cdot r_B G).$$

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

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

where

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

and sends it to entity *B*.

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

Key token processing (B2) Entity *B* verifies entity *A*'s signature on the key token KT_{A2} using entity *A*'s public verification key. If a signature scheme with message recovery is used, this includes recovering the data block DB_2 from the signature and verifying that entity *B*'s distinguishing identifier and the values $r_A G$ and $r_B G$ are contained in it. If a signature scheme with appendix is used, this includes reconstructing the data block DB_2 using the values in KT_{A1} and KT_{B1} and entity *B*'s distinguishing identifier and verifying the signature on that data block.

If the checks are successful, entity *B* verifies the $MAC_{K_{AB}}(DB_2)$ using the shared key

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

NOTES - This key agreement mechanism has the following properties:

1) Number of passes: *3*.

2) The mechanism provides mutual forward secrecy.

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

D.9 The Full Unified Model

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

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

— for each entity *X*, a private key-agreement key d_X, which is an integer in the range {*2,...,n-2*}, and a public key-agreement key P_X, which is an elliptic curve point satisfying $P_X = d_X G$.

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

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key Token Construction (A1) Entity *A* randomly and secretly generates r_A in the range {2,...,*n-2*}, computes $r_A G$, constructs the key token

$$KT_{A1} = r_A G$$

and sends it to entity *B*.

Key Token Construction (B1) Entity *B* randomly and secretly generates r_B in the range {2,...,*n-2*}, computes $r_B G$, constructs the key token

$$KT_{B1} = r_B G$$

and sends it to entity *A*.

Key Construction (A2) Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *A* computes the shared key

$$K_{AB} = (r_A \cdot l)(j \cdot KT_{B1}) \,||\, (d_A \cdot l)(j \cdot P_B).$$

Key Construction (B2) Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *B* computes the shared key

$$K_{AB} = (r_B \cdot l)(j \cdot KT_{A1}) \,||\, (d_B \cdot l)(j \cdot P_A).$$

NOTES - This key agreement mechanism has the following properties:

1) Number of passes: *2*.

2) The mechanism provides mutual forward secrecy.

3) The mechanism provides mutual implicit key authentication.

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

D.10 Key agreement of MQV type with 1 pass

This [15] is an example of key agreement mechanism 8. This key agreement mechanism establishes a shared secret between two entities *A* and *B* in one pass.

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

— for each entity *X*, a private key-agreement key d_X and a public key-agreement key P_X, which is an elliptic curve point satisfying $P_X = d_X G$. See ISO/IEC 15946-1 for a description of how to generate this key pair.

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

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key token construction (A1.1) Entity *A* randomly and secretly generates *r* in the range {*2,...,n-2*}, computes *rG*, constructs the key token,

$$KT_{A1} = rG$$

and sends it to entity *B*.

Key construction (A1.2) Entity *A* computes the shared key

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

Key construction (B1) Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Using its own private key, entity *B* computes the shared key from KT_{A1} as follows:

$$\mathrm{K_{AB}} = ((\mathrm{d_B} + \pi(\mathrm{P_B})\mathrm{d_B}) \cdot \mathrm{l})(\mathrm{j} \cdot (\mathrm{KT_{A1}} + \pi(\mathrm{KT_{A1}})\mathrm{P_A})).$$

NOTES - This key agreement mechanism has the following properties:

1) Number of passes: *1*.

2) The mechanism provides mutual implicit key authentication.

3) The mechanism provides forward secrecy with respect to entity *A*.

D.11 Key agreement of MQV type with 2 passes

This [15] is an example of key agreement mechanism 9. This key agreement mechanism establishes a shared secret between entities *A* and *B* in two passes.

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

— for each entity *X*, a private key-agreement key d_X and a public key-agreement key P_X, which is an elliptic curve point satisfying $P_X = d_XG$. See ISO/IEC 15946-1 for a description of how to generate this key pair.

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

Each entity should independently verify that the other entity's public key is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this.

The values *l* and *j* are used for cofactor multiplication as explained in Clause 7.

Key token construction (A1) Entity *A* randomly and secretly generates r_A in the range {2,...,*n*-2}, computes r_AG, constructs the key token,

$$\mathrm{KT_{A1}} = \mathrm{r_A G}$$

and sends it to entity *B*.

Key token construction (B1) Entity *B* randomly and secretly generates r_B in the range {2,...,n-2}, computes r_BG, constructs the key token,

$$\mathrm{KT_{B1}} = \mathrm{r_B G}$$

and sends it to entity *A*.

Key construction (A2) Entity *A* should verify that KT_{B1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *A* computes the shared key

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

Key construction (B2) Entity *B* should verify that KT_{A1} is indeed a point on the elliptic curve. See ISO/IEC 15946-1 for a description of how to do this. Entity *B* computes the shared key

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