INTERNATIONAL STANDARD

ISO/IEC 11770-3

First edition
1999-11-01

Information technology — Security techniques — Key management —

Part 3: Mechanisms using asymmetric techniques

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

Partie 3: Mécanismes utilisant des techniques asymétriques

Reference number
ISO/IEC 11770-3:1999(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Foreword

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

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

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

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 11770 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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

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: Mecanisms using asymmetruc techniques*

Further parts may follow.

Annexes A to E of this part of ISO/IEC 11770 are for information only.

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 an 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.

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 normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 11770. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 11770 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 7498-2:1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture.*

ISO/IEC 9594-8:1995, *Information technology - Open Systems Interconnection – The Directory: Authentication framework.*

ISO/IEC 9798-3:1998, *Information technology - Security techniques - Entity authentication - Part 3: Mechanisms using digital signature techniques.*

ISO/IEC 10118-1:1994, *Information technology - Security techniques - Hash-functions - Part 1: General.*

ISO/IEC 10181-1:1996, *Information technology - Open Systems Interconnection - Security frameworks for open systems Overview.*

ISO/IEC 11770-1:1996, *Information technology - Security techniques - Key management - Part 1: Framework.*

3. Definitions

For the purposes of this part of ISO/IEC 11770, the following definitions apply.

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

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

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

3.4. certification authority (CA): a center trusted to create and assign public key certificates. Optionally, the certification authority may create and assign keys to the entities.

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

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

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

3.8. digital signature: a data 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.

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

3.10. encipherment: the (reversible) transformation of data by a cryptographic algorithm to produce ciphertext, i.e. to hide the information content of the data [ISO/IEC 11770-1:1996].

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.24. private key: that key of an entity's asymmetric key pair which 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.

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

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

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

3.27. public key information: information containing at least the entity's distinguishing identifier and public key. The public key information is limited to data regarding one entity, and one public key for this entity. There may be other static information regarding the certification authority, the entity, the public key, restrictions on key usage, the validity period, or the involved algorithms, included in the public key information.

3.28. secret key: a key used with symmetric cryptographic techniques by a specified set of entities.

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

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

3.31. time stamp: a data item which denotes a point in time with respect to a common time reference.

3.32. time stamping authority: a trusted third party trusted to provide evidence which includes the time when the secure time stamp is generated [ISO/IEC 13888-1:1997].

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

3.34. trusted third party: a security authority, or its agent, trusted by other entities with respect to security related activities [ISO/IEC 10181-1:1996].

4. Symbols and abbreviations

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

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_A	entity *A*'s public encipherment transformation.
e_A	entity *A*'s public encipherment key.
F(h,g)	the key agreement function.
f	cryptographic check function
$f_K(Z)$	cryptographic check value which is the result of applying the cryptographic check function *f* using as input a secret key *K* and an arbitrary data string *Z*.
g	the common element shared publicly by all the entities that use the key agreement function *F*.
h_A	entity *A*'s private key agreement key.
hash	hash-function
H	set of elements
G	set of elements
K	a secret key for a symmetric cryptosystem.
K_{AB}	a secret key shared between entities *A* and *B*.

NOTE - In practical implementations the shared secret key may be subject to further processing before it can be used for a symmetric cryptosystem.

KT	key token.
KT_{Ai}	the key token sent by entity *A* after processing phase i.
p_A	entity *A*'s public key agreement key.
PKI_A	entity *A*'s public key information
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_A	entity *A*'s private signature transformation.
s_A	entity *A*'s private signature key.
Texti	an optional data field whose use is beyond the scope of this part of ISO/IEC 11770.
TVP	time-variant parameter, such as a random number, a time stamp, or a sequence number.
V_A	entity *A*'s public verification transformation.
v_A	entity *A*'s public verification key.
w	one-way function
Σ	the digital signature
||	concatenation of two data elements.

NOTES

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 signature Σ.

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 inclu-

sion 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 Part 1 for registration of key).

6. 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 G, a set H and a function F. The function F shall satisfy the following requirements:

1. F operates on two inputs, one element h from H and one element g from G, and produces a result y in G, $y = F(h,g)$.
2. F satisfies the commutativity condition $F(h_A,F(h_B,g)) = F(h_B,F(h_A,g))$.
3. 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.
4. The entities *A* and *B* share a common element g in G which may be publicly known.
5. 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. An example of a possible function F is given in Annex B.
2. In practical implementations of the key agreement mechanisms the shared secret key may be subject to further processing. A derived shared secret key may 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 Diffie-Hellman key agreement is given in clause B.5.

6.1 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 8.

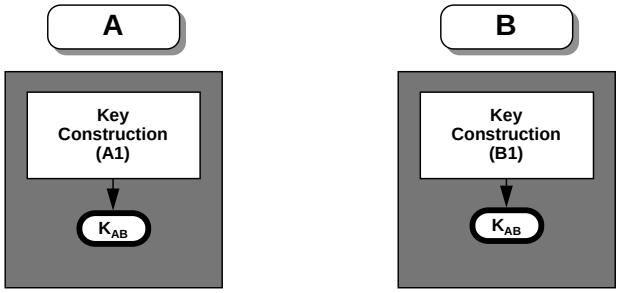

Figure 1 - Key Agreement Mechanism 1

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

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

Key Construction (B1) *B* computes, using its own private key agreement key h_B and *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, the two computed values for the key K_{AB} are identical.

NOTE - 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 6 note 2).

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

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 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: an example known as Diffie-Hellman key agreement is given in clause B.5.

6.2 Key agreement mechanism 2

This key agreement mechanism establishes in one pass a shared secret key between A and B with implicit key authentication from B to A, but no entity authentication from A to B (i.e. 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 B's public key agreement key p_B. This may be achieved using the mechanisms of clause 8.

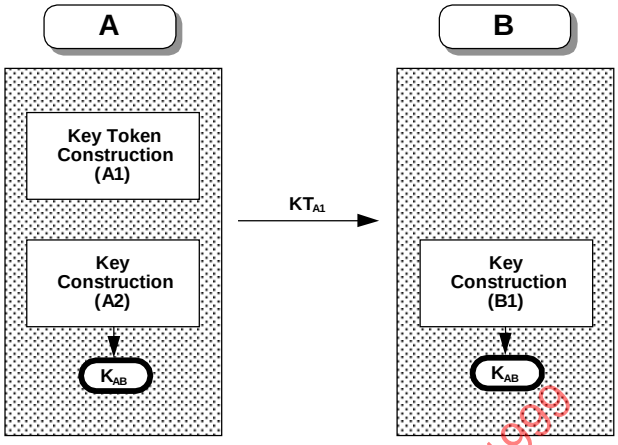

Figure 2 - Key Agreement Mechanism 2

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

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

to B.

Key Construction (A2) Further A computes the key as

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

Key Construction (B1) 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, the two computed values for the key K_{AB} are identical.

NOTE - This Key Agreement Mechanism has the following properties:

1. Number of passes: 1.

2. Key authentication: this mechanism provides implicit key authentication from B to A (B is the only entity other than 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 A and B's private key agreement key. However, since entity A may 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 *B*'s public key and sending KT_{A1}.

5. Example: an example of this key agreement mechanism (known as ElGamal key agreement) is described in clause B.3.

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

6.3 Key agreement mechanism 3

This key agreement mechanism establishes in one pass a shared secret key between *A* and *B* with mutual implicit key authentication, and entity authentication of *A* to *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 8.
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 8.
5. *TVP:* 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 cryptographic check function *f* (such as those standardized in ISO/IEC 9797) and a way to incorporate K_{AB} as the key in this check function.

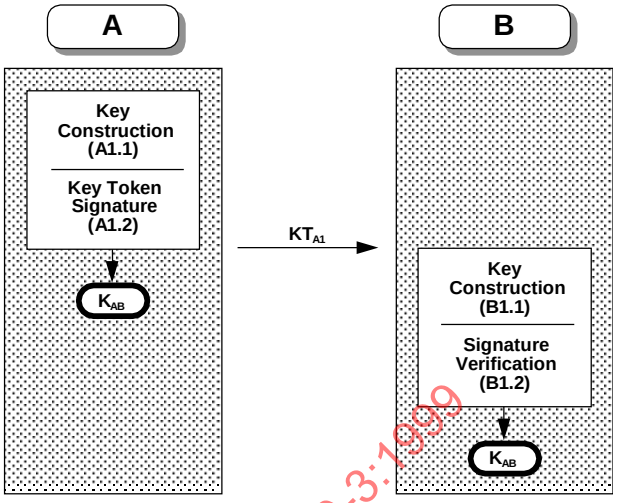

Figure 3 - Key Agreement Mechanism 3

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

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

Using the shared secret key K_{AB}. *A* computes a cryptographic check value on the concatenation of the sender's distinguishing identifier *A* and a sequence number or time stamp *TVP*.

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

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

and sends it to *B*.

Key Construction (B1.1) *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} *B* computes the cryptographic check value on the sender's distinguishing identifier *A* and the *TVP*.

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

NOTE - This Key Agreement Mechanism has the following properties:

1. Number of passes: 1.

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

3. Key confirmation: this mechanism provides key confirmation from *A* to *B*.

4. This is a key agreement mechanism since the established key is a one-way function of a random value *r* supplied by *A* and *B*'s private key agreement key. However, since entity *A* may 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 *B*'s public key and sending KT_{A1}.

5. *TVP*: provides entity authentication of *A* to *B* and prevents replay of the key token.

6. Example: an example of this key agreement mechanism (known as Nyberg-Rueppel key agreement) is described in clause B.4.

7. Public key certificates: if *Text1* is used to transfer *A*'s public key certificate, then requirement 2 at the beginning of this clause can be relaxed to the requirement that *B* is in possession of an authenticated copy of the CA's public verification key.

6.4 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.

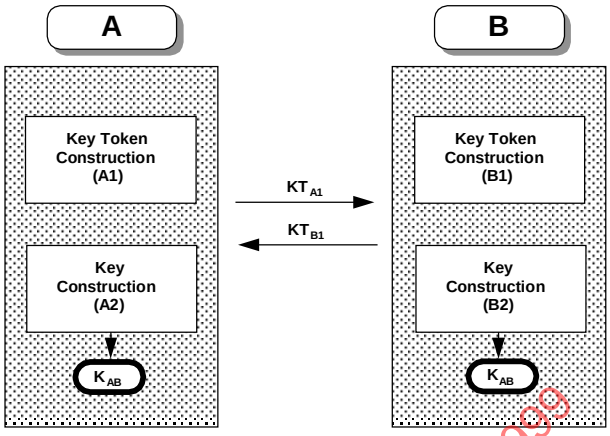

Figure 4 - Key Agreement Mechanism 4

Key Token Construction (A1) *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 *B*.

Key Token Construction (B1) *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 *A*.

Key Construction (A2) *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) *B* extracts $F(r_A,g)$ from the received key token KT_{A1} and computes the shared secret key

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

NOTE - 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. Key confirmation: this mechanism provides no key confirmation.

3. 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 A and B respectively. However, since entity B may 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}.

4. Example: an example of this mechanism (known as Diffie-Hellman key agreement) is described in clause B.5.

6.5 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 8.
3. Both entities have agreed on a common one-way function w.

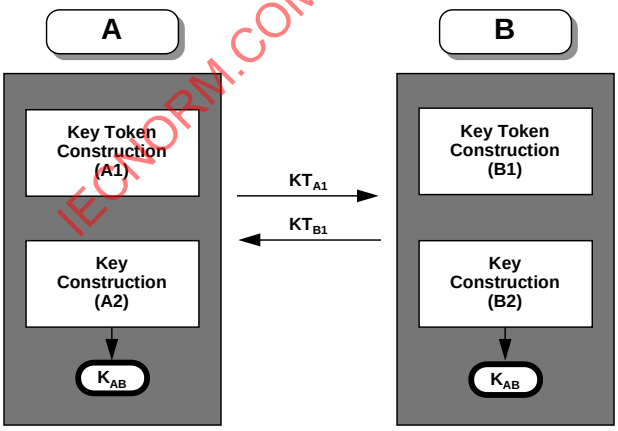

Figure 5 - Key Agreement Mechanism 5

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

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

to B.

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

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

to A.

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

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

where w is a one-way function.

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

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

NOTE - 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 cryptographic check value (on known data) computed using the key K_{AB}, then this mechanism provides explicit key authentication from B to A.

3. Key confirmation: if the data field *Text2* contains a cryptographic check value (on known data) computed using the key K_{AB}, then this mechanism provides key confirmation from B to 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 A and B respectively. However, since entity B may 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}.

5. Example: An example of this key agreement mechanism (known as the Matsumoto-Takashima-Imai A(0) key agreement scheme) is described in

clause B.6. Another example is known as the Goss protocol.

6. The function w has to hide its inputs in the sense that from the function value and from one of the inputs it is infeasible to compute the relevant part of the other input. This may be achieved by using a hash-function from ISO/IEC 10118 (there is no need for a collision-resistant hash-function).

7. 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.

6.6 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. *A* has an asymmetric encipherment system with the transformations (E_A,D_A).
2. *B* has an asymmetric signature system with the transformations (S_B,V_B).
3. *A* has access to an authenticated copy of *B*'s public verification transformation V_B. This may be achieved using the mechanisms of clause 8.
4. *B* has access to an authenticated copy of *A*'s public encipherment transformation E_A. This may be achieved using the mechanisms of clause 8.

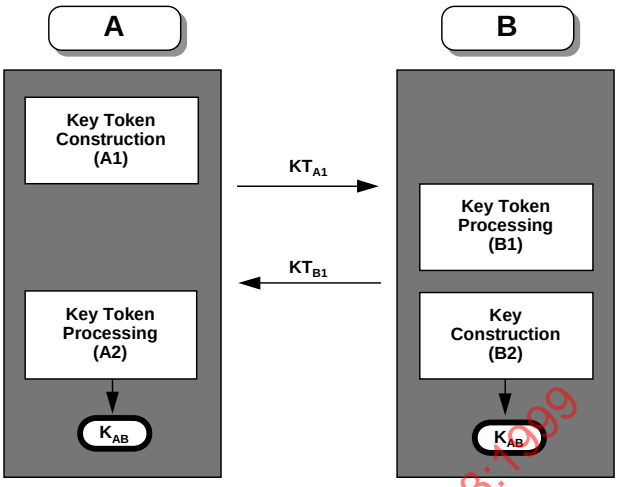

Figure 6 - Key Agreement Mechanism 6

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

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

to *B*.

Key Token Processing (B1) *B* generates a random number r_B and signs a data block consisting of the distinguishing identifier *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 B enciphers a data block consisting of its distinguishing identifier *B* (optional), the signed block *BS* and some optional data *Text3*, using *A*'s public encipherment transformation E_A, and sends the key token

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

back to *A*.

Key Construction (B2) The shared secret key consists of all or part of *B*'s signature Σ contained in the signed block *BS* (see Notes 1 in clause 4).

Key Token Processing (A2) *A* deciphers the key token KT_{B1} using its private decipherment transformation D_A, optionally checks the sender identifier *B*, and uses *B*'s public verification transformation V_B to verify the digital signature of the signed block *BS*. Then *A* checks the recipient identifier *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 success-

ful, *A* accepts all or part of *B*'s signature Σ of the signed block *BS* as the shared secret key.

NOTE - This Key Agreement Mechanism has the following properties:

1. Number of passes: 2.

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

3. Key confirmation: If the data field *Text3* contains a cryptographic check value (on known data) computed using the key K_{AB}, then this mechanism provides key confirmation from *B* to *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 *A* and *B* respectively. However, since entity *B* may 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}.

5. Example: this mechanism is derived from Beller and Yacobi's two pass protocol described in clause B.7.

6. Public key certificates: if *Text1* and *Text4* contain the public key certificate of entity A's encipherment key and the public key certificate of 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.

7. A significant feature of this scheme is that the identity of party B may remain anonymous to eavesdroppers, of particular advantage in the wireless environment which is a main environment for the application of this scheme.

6.7 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 8.
3. Each entity has a common cryptographic check function *f*.

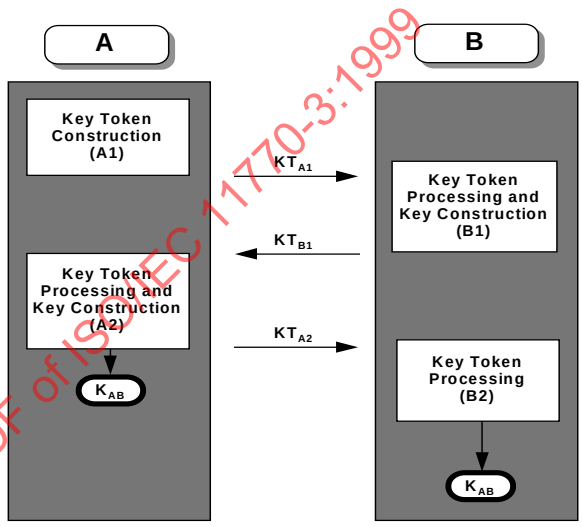

Figure 7 - Key Agreement Mechanism 7

Key Token Construction (A1) *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 *B*.

Key Token Processing and Key Construction (B1) *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

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

where

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

and sends it back to *A*.

Key confirmation is provided by sending $f_{K_{AB}}(DB_1)$ in KT_{B1}. Alternatively, if both parties have a common symmetric encryption system, key confirmation can be

obtained by encrypting part of the token as follows: replace KT_{B1} by $F(r_B,g)$ followed by $E_{K_{AB}}(S_B(DB_1))$.

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

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

Using K_{AB}, *A* verifies the cryptographic check value $f_{K_{AB}}(DB_1)$.

Then *A* constructs the signed key token

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

where

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

and sends it to *B*.

Key confirmation is provided by sending $f_{K_{AB}}(DB_2)$ in KT_{A2}. Alternatively, key confirmation can be obtained by encrypting part of the token as follows: replace KT_{A2} by $E_{K_{AB}}(S_A(DB_2))$.

Key Token Processing (B2) *B* verifies *A*'s signature on the key token KT_{A2}, using *A*'s public verification key, then verifies *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, *B* verifies the cryptographic check value $f_{K_{AB}}(DB_2)$ using

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

NOTE - 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 *A* and *B* respectively. However, since entity *B* may 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}.

5. Example: an example of this key agreement mechanism may be provided by the Diffie-Hellman scheme described in Annex B 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. 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, 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, 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 entity *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.

7. 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 center), 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 xor-ed by a (pseudo-) random bit pattern to destroy any apparent structure in the key data block.

7.1. 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 *B* to *A*. The following requirements shall be satisfied:

1. Entity *B* has an asymmetric encipherment system (E_B, D_B).
2. A has access to an authenticated copy of B's public encipherment transformation E_B. This may be achieved using the mechanisms of clause 8.
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 *A* and *B* have to maintain bilateral counters.

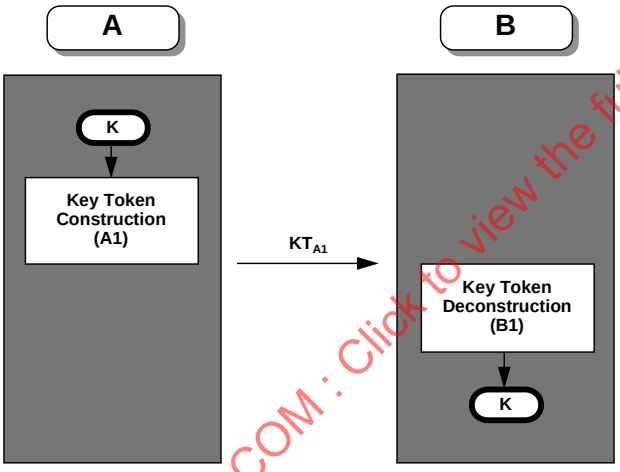

Figure 8 - Key Transport Mechanism 1

Key Token Construction (A1) *A* has obtained a key *K* and wants to transfer it securely to *B*. *A* constructs a key data block consisting of its distinguishing identifier *A* (optional), the key *K*, an optional *TVP* and an optional data field *Text1*. Then *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 *B*.

Key Token Deconstruction (B1) *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 *A*.

NOTE - This Key Transport Mechanism has the following properties:

1. Number of passes: 1.
2. Key authentication: this mechanism provides implicit key authentication from *B* to *A* since only *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 *B* receives the key *K* from the non-authenticated entity *A*, secure usage of *K* by *B* is restricted to functions not requiring trust in 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 clause B.8. Another example of this mechanism using RSA is described in clause B.10.

7.2. 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 implicit key authentication from *A* to *B*. 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 *B*'s public encipherment transformation E_B. This may be achieved using the mechanisms of clause 8.
4. Entity *B* has access to an authenticated copy of *A*'s public verification transformation V_A. This may be achieved using the mechanisms of clause 8.

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 *A* and *B* have to maintain bilateral counters.

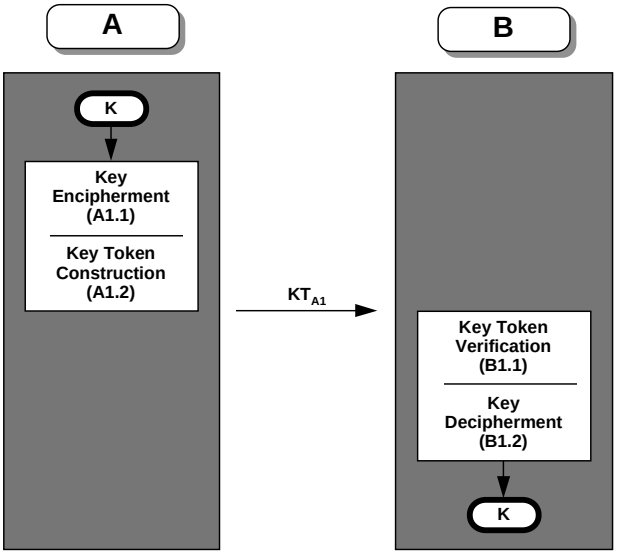

Figure 9 - Key Transport Mechanism 2

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

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

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

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

to *B*.

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

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

NOTE - This Key Transport Mechanism has the following properties:

1. Number of passes: 1.

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

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

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

5. *TVP* (optional): provides entity authentication of *A* to *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*.

6. Data field *A*: *A*'s distinguishing identifier is included in the enciphered block *BE* to prevent *A* from misappropriating an enciphered key block intended for use by another entity. This is achieved by comparing *A*'s identity with *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. 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 *A* to *B* and from *B* to *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 non-confidential elements as well as the enciphered keys.

11. Examples of this mechanism are described in clauses B.9 and C.7.

7.3. 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 *B*'s public encipherment transformation E_B. This may be achieved using the mechanisms of clause 8.
4. Entity *B* has access to an authenticated copy of *A*'s public verification transformation V_A. This may be achieved using the mechanisms of clause 8.
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 *A* and *B* have to maintain bilateral counters.

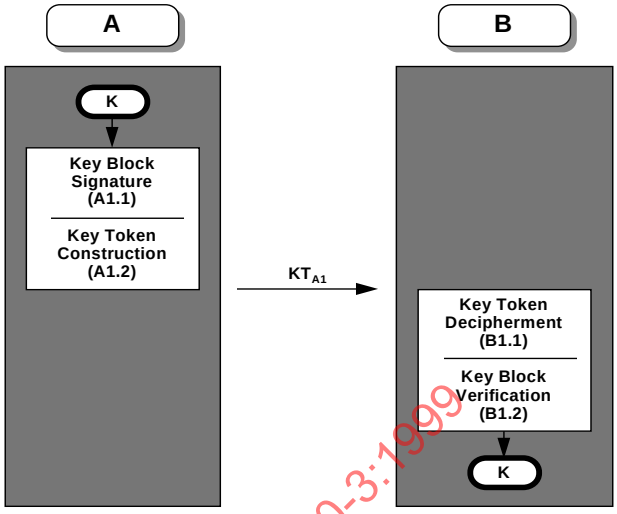

Figure 10 - Key Transport Mechanism 3

Key Block Signature (A1.1) *A* has obtained a key *K* and wants to transfer it securely to *B*. *A* forms a key data block consisting of the recipient's distinguishing identifier *B*, the key *K*, an optional sequence number or time stamp *TVP*, and some optional data. Then 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) *A* forms the token data block, consisting of the signed block *BS* and some optional *Text2*. Then *A* enciphers the token data block using the receiver's public encipherment transformation E_B and sends the resulting key token

$$KT_{A1} = E_B(BS||\ Text2)||Text3$$

to *B*.

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

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

NOTE - 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 *A* to *B* if the optional *TVP* is used, and implicit key authentication from *B* to *A*.

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

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

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

6. Data field *B*: *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 *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 *A* to *B* and from *B* to *A*) then mutual entity authentication and joint key control can be provided (depending on the use of the optional *TVP*).

7.4. 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 *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 *B*'s public verification transformation V_B. This may be achieved using the mechanisms of clause 8.
4. Entity *B* has access to an authenticated copy of *A*'s public encipherment transformation E_A. This may be achieved using the mechanisms of clause 8.

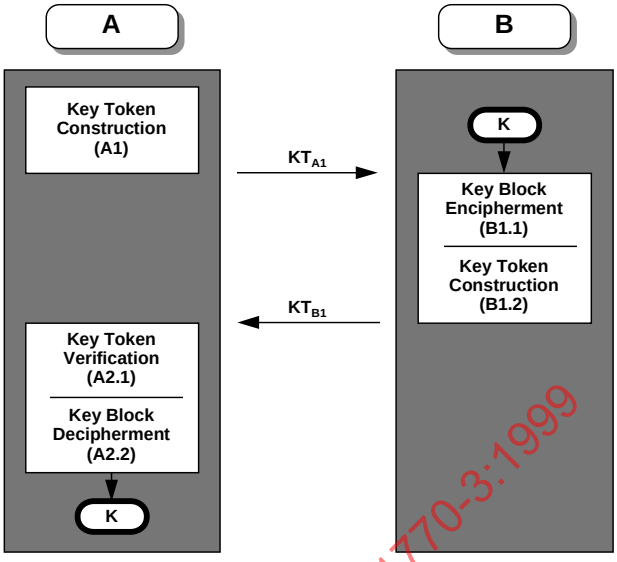

Figure 11 - Key Transport Mechanism 4

Key Token Construction (A1) *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 *B*.

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

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

Key Token Construction (B1.2) *B* forms the token data block, consisting of the recipient's distinguishing identifier *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 *B* signs the token data block with its private signature transformation S_B and sends the resulting key token

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

to *A*.

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

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

NOTE - 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 *B* to *A* and implicit key authentication from *A* to *B*.

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

4. Key control: *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 (note that the roles of *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 *A* to *B* and from *B* to *A*) then mutual entity authentication and joint key control can be provided.

7.5. 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 *A* to *B* and one key from *B* to *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 8.
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 8.

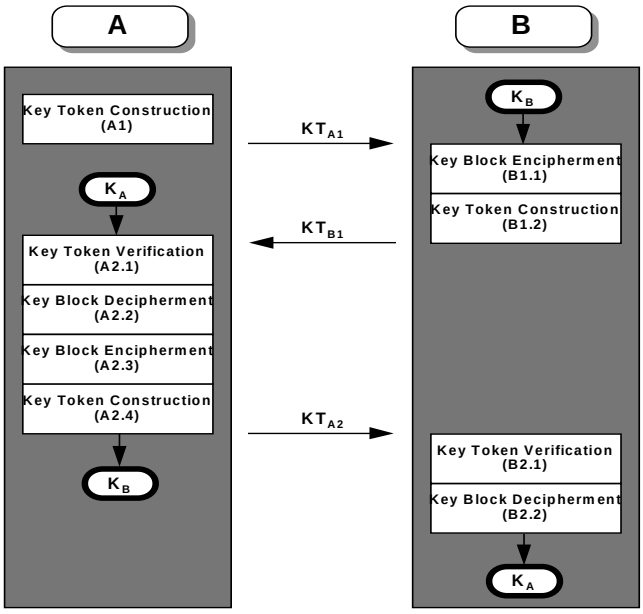

Figure 12 - Key Transport Mechanism 5

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

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

and sends it to *B*.

Key Block Encipherment (B1.1) *B* has obtained a key K_B and wants to transfer it securely to *A*. *B* constructs a block containing its own distinguishing identifier *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 *B* randomly generates r_B and constructs a data block, containing r_B, r_A, the recipient's identity *A*, the enciphered key block BE_1, and some optional *Text3*. *B* signs the block using its private signature transformation S_B, and sends the key token

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

to *A*.

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

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

Key Block Encipherment (A2.3) Then *A* constructs a data block, containing its own distinguishing identifier *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 *A* constructs a data block, containing the random number r_A, the random number r_B, the recipient's distinguishing identifier *B*, the enciphered key block BE_2, and some optional *Text6*. *A* signs the data block using its private signature transformation S_A, and sends the key token

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

to *B*.

Key Token Verification (B2.1) *B* verifies *A*'s signature on the key token KT_{A2}, using *A*'s public verification transformation V_A, checks the distinguishing identifier *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) *B* deciphers the enciphered block BE_2 using its private decipherment transformation D_B and verifies the distinguishing identifier *A*. If all checks are successful, *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.

NOTE - 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 *B* to *A* and implicit key authentication of K_B from *A* to *B*.

3. Key confirmation: this mechanism provides key confirmation from sender to recipient for both keys K_A and K_B. Moreover, if *A* includes a cryptographic check value 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: *A* can choose the key K_A, since it is the originating entity. Similarly, *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 *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. 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 cryptographic check value within *Text6*.

6. Public key certificates: if the data fields *Text1* and *Text4* (or *Text7* and *Text4*) each contain the public key certificates of entity *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.

7.6. Key transport mechanism 6

This key transport mechanism securely transfers in three passes two secret keys, one from *A* to *B* and one from *B* to *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 8.

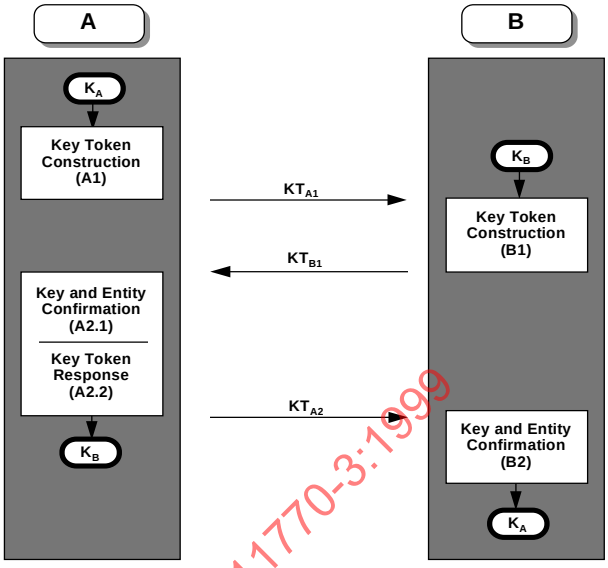

Figure 13 - Key Transport Mechanism 6

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

$$BE_1 = E_B(A||K_A||r_A||Text1)$$

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

$$KT_{A1} = BE_1||Text2$$

A sends the token to *B*.

Key Token Construction (B1) *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 *B* verifies the sender identity *A*.

B has obtained a key K_B and wants to transfer it securely to *A*. *B* selects a random number r_B and constructs a key data block consisting of the distinguishing identifier *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 *B* enciphers the key block using *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 B constructs the key token KT_{B1}, consisting of the enciphered data block BE_2 and an optional data field *Text4*,

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

B sends the token to A.

Key and Entity Confirmation (A2.1) 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 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, A has authenticated B and at the same time obtained confirmation that K_A has safely reached entity B.

Key Token Response (A2.2) 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*,

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

A sends the token to B.

Key and Entity Confirmation (B2) 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, B has authenticated A and at the same time has obtained confirmation that K_B has safely reached entity B.

NOTE - 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 B to A and implicit key authentication of K_B from A to B..
3. Key confirmation: this mechanism provides mutual key confirmation.
4. Key control: A can choose the key K_A, since it is the originating entity. Similarly, 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 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 A to B and K_B from B to A. The following cryptographic function separation may be derived from the mechanism: A uses its key K_A to encipher messages for B and to verify authentication codes from B. B in turn uses the received key K_A to decipher messages from A and generate authentication codes for 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 (see Brandt et al. in the Bibliography).
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.

8. 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 A. The public key information includes at least A's distinguishing identifier and A's public key.

8.1. 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.

8.1.1 Public key transport mechanism 1

If *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 *B* then *A* may transport its public key information directly via that protected channel to *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 *A*'s distinguishing identifier and *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.

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

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

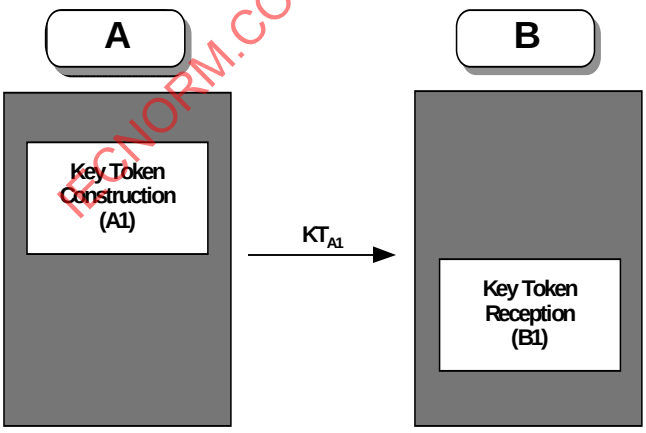

Figure 14 - Public Key Transport Mechanism 1

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

NOTE - 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).

8.1.2 Public key transport mechanism 2

This mechanism transports the public key information of entity *A* via an unprotected channel to *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, registered mail. As an additional requirement the entities shall share a common hash-function *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 *A*'s distinguishing identifier and *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.

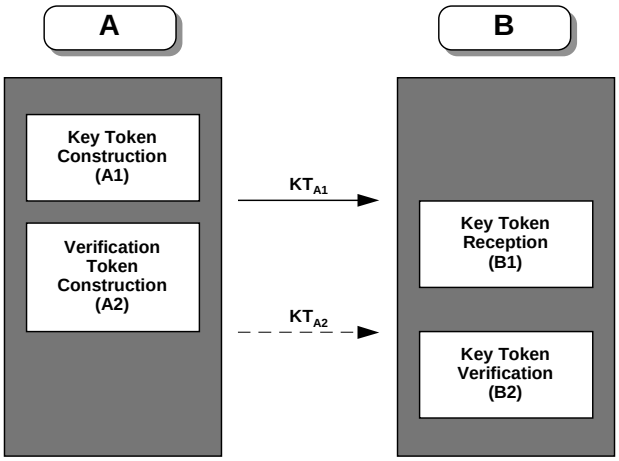

Figure 15 - Public Key Transport Mechanism 2

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

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

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

Verification Token Construction (A2) *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 *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 *A* and *B*, computes the check value on the public key information of *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, *B* puts *A*'s public key into the list of active public keys (this list shall be protected from tampering).

NOTE - 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 *A* may sign the token KT_{A1} using the corresponding private signature key. In that case, the verification of *A*'s signature in step (B1) using the received public verification key confirms that *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 *A* currently knows the corresponding private signature key.

4. A manually signed letter may be used for the verification token.

8.2. 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. A 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, 11770-1 (Annex D).

8.2.1 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 *A*'s public key information PKI_A has been issued by some certification authority, and that *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.

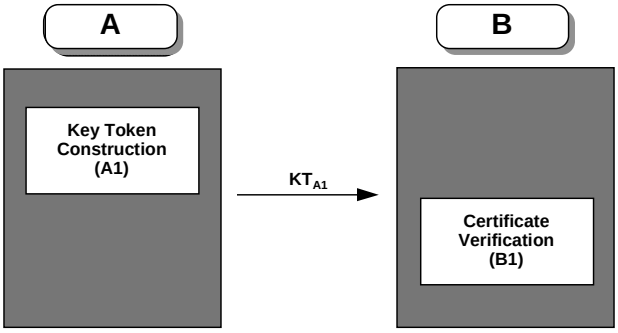

Figure 16 - Public Key Transport Mechanism 3

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

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

Certificate Verification (B1) Upon reception of the public key certificate, *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 *A*'s public key.

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

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

1. Number of passes: 1. But there may have been a request from *B* to *A* for the transfer of the public key certificate. This additional pass is optional and not shown here. *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 *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 *B* in an authenticated way. This can be done using the mechanisms described in clause 8.

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 the mechanism provides the property with respect to entity *A*.

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.

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 *B* needs one computation of the function *F* in Key Agreement Mechanism 2.

Properties of Key Agreement Mechanisms:

Mechanism	1	2	3	4	5	6	7
Number of passes	0	1	1	2	2	2	3
Implicit key authentication	*A*,*B*	*B*	*A*,*B*	no	*A*,*B*	*A*,*B*	*A*,*B*
Key confirmation	no	no	B	no	opt	opt	*A*,*B*
Entity authentication	no	no	(*A*)	no	no	*B*	*A*,*B*
Public key operations	1,1	2,1	3 (or 2),2	2,2	2,2	2,2	3,3

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	4,4	4,4	2,2

Annex B
(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 G and H, which is conjectured to satisfy the five properties listed in clause 6, given that certain parameters are chosen appropriately.

Let p be a prime number, G be the set of elements of the Galois field with p elements, F_p, and let $H = \{1, \ldots, p\text{-}2\}$. 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 $F(\cdot,g)$ 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 X, and a public key $p_X = g^{h_X} \bmod p$ 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 $p\text{-}1$ 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.

B.1. Non-interactive Diffie-Hellman key agreement

This [6] is an example of Key Agreement Mechanism 1.

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

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

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

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

B.2. Identity-based mechanism

This [8] 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, Annex A (informative). 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($p\text{-}1$, $q\text{-}1$).

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 X and possibly a serial number, a validity period, a time

stamp and other data elements. Then X's key management pair is (h_X, p_X) where h_X is an integer less than n and

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

Its certificate is computed by the certification authority as

$$Cert_X = s_X O^{h_X} \pmod{n},$$

where s_X is the integer such that:

$$s_X{}^y I_X = 1 \pmod{n}$$

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

$$p_B = Cert_B{}^y . I_B \mod n$$

and computes the shared secret key as

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

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

$$p_A = Cert_A{}^y . I_A \mod n$$

and computes the shared secret key as

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

NOTE - A one-pass and a two-pass identity-based mechanisms using the same set-up are described in the references [8], [19] and [20] of the Annex D (Bibliography).

B.3. ElGamal key agreement

This [7] 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) A randomly and secretly generates r in $\{1, \ldots, p-2\}$, computes $g^r \bmod p$ and constructs the key token

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

and sends it to B.

Key Construction (A2) A computes the shared key

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

Key Construction (B1) B computes the shared key

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

B.4. Nyberg-Rueppel key agreement

This [18] 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 $H = \{1, \ldots, q-1\}$. Then 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 $\{1, \ldots, p-1\}$, for example, in H.

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

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

Further A computes the shared secret key as

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

Using the shared secret key K_{AB} A computes a cryptographic check value on the sender's distinguished identifier A and a sequence number or time-stamp *TVP*.

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

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

$$y = r - h_A e' \mod q$$

A forms the key token

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

and sends it to *B*.

Key Construction (B1.1) *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} *B* computes the cryptographic check value on the sender's distinguished identifier *A* and the *TVP*, and computes

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

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

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

B.5. Diffie-Hellman key agreement

This [6] 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) *A* randomly and secretly generates r_A in $\{1, \ldots, p-2\}$, computes $g^{r_A} \bmod p$, constructs the key token

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

and sends it to *B*.

Key Token Construction (B1) *B* randomly and secretly generates r_B in $\{1, \ldots, p-2\}$, computes $g^{r_B} \bmod p$, constructs the key token

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

and sends it to *A*.

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

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

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

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

B.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) *A* randomly and secretly generates r_A in $\{1, \ldots, p-2\}$, computes the key token

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

and sends it to *B*.

Key Token Construction (B1) *B* randomly and secretly generates r_B in $\{1, \ldots, p-2\}$, computes the key token

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

and sends it to *A*.

Key Construction (B2) *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) *A* computes the shared key as

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

B.7. Beller-Yacobi Protocol

This part of the Annex gives a description of the original Beller-Yacobi protocol [4], 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 B's signature verification key and its certificate to A in a confidential way, thus assuring anonymity.

Let *enc: K□M → C* be a conventional encryption function, such as DES, 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 EG 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: *B* picks a random number r_B and computes

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

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

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

and sends it to *B*.

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

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

Then *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 *A*.

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

Entity Authentication and Key Construction (A2) *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 party *B*. *A* verifies certificate C_B, and if positive it then uses the verification function, V_B to verify *B*'s signature *BS*. If positive it then accepts *u* as a shared secret key.

B.8. ElGamal key transfer

This [7] is an example of Key Transport Mechanism 1. An appropriate prime *p* and generator *g* of Z_p^* are selected and made public. *B*'s private and public key agreement keys are, respectively, h_B and

$$p_B = g^{h_B} \bmod p$$

Key Token Construction (A1) *A* has obtained a key *K* (in the range *0<K<p*) and wants to transfer it securely to *B*. *A* randomly and secretly generates a random integer *r*, *1 < r < p-1*, and enciphers *K* as

$$BE = K \cdot (p_B)^r \bmod p$$

Then *A* constructs the key token

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

and sends it to *B*.

Key Token Deconstruction (B1) *B* recovers the key *K* using its private key agreement key h_B, computing

$$K = BE \cdot (g^r)^{-h_B} \bmod p$$

B.9. ElGamal key transfer with originator's signature

This is an example of Key Transport Mechanism 2. An appropriate prime *p* and generator *g* of Z_p^* are selected and made public. *B*'s private and public key agreement keys are, respectively, h_B and

$$p_B = g^{h_B} \bmod p$$

A's private and public signature transformations are respectively denoted S_A and V_A; (S_A, V_A) could denote any signature system, for example RSA signature and signature verification as defined in ISO/IEC 9796.

Key Encipherment (A1.1) *A* has obtained a key *K* and wants to transfer it securely to *B*. *A* randomly and secretly generates a random integer *r*, in *{1,…,p-2}* and enciphers the key data block *A||K* as

$$BE = (A \| K) \cdot (p_B)^r \bmod p$$

Note that *K* must be chosen in such a way that the value of *(A||K)* is less than the prime *p*.

Key Token Construction (A1.2) *A* forms the token data block, consisting of the recipient's distinguished identifier *B*, an optional time stamp or sequence number *TVP*, g^r and the enciphered block *BE*. Then *A* signs