INTERNATIONAL STANDARD

ISO/IEC 11770-5

First edition
2011-12-15

Information technology — Security techniques — Key management — Part 5: Group key management

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

Partie 5: Gestion de clés de groupe

Reference number
ISO/IEC 11770-5:2011(E)

Contents

Page

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.

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

ISO/IEC 11770-5 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: Mechanisms using asymmetric techniques*

— *Part 4: Mechanisms based on weak secrets*

— *Part 5: Group key management*

Introduction

This part of ISO/IEC 11770 does not specify the means to be used to establish initial secret keys; that is, all the mechanisms specified in this part of ISO/IEC 11770 require an entity to share the secret key with another entity, the key distribution centre (KDC). For general guidance on the key lifecycle see ISO/IEC 11770-1. This part of ISO/IEC 11770 does not explicitly address the issue of interdomain key management. This part of ISO/IEC 11770 also does not define the implementation of key establishment mechanisms; products complying with this part of ISO/IEC 11770 might be compatible.

This part of ISO/IEC 11770 does not specify the information which has no relation with key establishment mechanisms, nor does it specify other messages such as error messages. The explicit format of messages is not within the scope of this part of ISO/IEC 11770.

The mechanisms specified in this part of ISO/IEC 11770 have been assigned object identifiers in accordance with ISO/IEC 9834. The list of assigned object identifiers is given in the normative Annex A. Any change to the specification of the mechanisms resulting in a change of functional behavior will result in a change of the object identifier assigned to the mechanisms.

Information technology — Security techniques — Key management —

Part 5: Group key management

1 Scope

This part of ISO/IEC 11770 specifies key establishment mechanisms for multiple entities to provide procedures for handling cryptographic keying material used in symmetric or asymmetric cryptographic algorithms according to the security policy in force.

It defines symmetric key based key establishment mechanisms for multiple entities with a key distribution centre (KDC), and defines symmetric key establishment mechanisms based on a general tree based structure with both individual rekeying and batched rekeying. It also defines key establishment mechanisms based on a key chain with both unlimited forward key chain and limited forward key chain. The two types of key establishment mechanisms can be combined by applications.

This part of ISO/IEC 11770 also describes the required content of messages which carry keying material or are necessary to set up the conditions under which the keying material can be established.

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 10118-3:2004, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*

ISO/IEC 14888-2:2008, *Information technology — Security techniques — Digital signatures with appendix — Part 2: Integer factorization based mechanisms*

3 Terms and definitions

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

3.1
active
state of an entity in which the entity can obtain the shared secret key

3.2
ancestor keys of key *k*
set of keys in a logical key hierarchy that are assigned to the ancestor nodes of the node to which *k* is assigned

NOTE One of the keys in a set of ancestor keys is either the shared secret key or a key encryption key.

3.3
ancestor nodes of node *v*
set of nodes in a tree that can be reached by repeatedly going to the parent node from v

3.4
backward secrecy with interval *T*
security condition in which an entity joining at time $t = t_0$ cannot obtain any former shared secret keys at time $t < t_0 - T$

3.5
batch rekeying
rekeying method in which the shared secret key, and optionally, key encryption keys are updated at every rekeying interval T

3.6
child keys of key *k*
set of keys in a logical key hierarchy where the keys are assigned to the child nodes of the node to which k is assigned

NOTE One of the keys in a set of child keys shall be a key encryption key or individual key.

3.7
child nodes of node *w*
set of nodes in a tree which hang on w

3.8
***d*-ary tree**
tree where each node has d children except the leaf nodes in the tree

3.9
forward secrecy with interval *T*
security condition in which an entity leaving at time $t = t_0$ cannot obtain any subsequent shared secret keys at time $t > t_0 + T$

3.10
inactive
state of an entity in which the entity cannot obtain the shared secret key

3.11
individual key
key shared between the key distribution centre and each entity

3.12
individual rekeying
rekeying method in which the shared secret key, and optionally, key encryption keys are updated when an entity joins or leaves

3.13
key
sequence of symbols that controls the operations of a cryptographic transformation

3.14
key chain
set of cryptographic keys which are not necessarily independent

3.15
key distribution centre
KDC
entity trusted to generate or acquire, and distribute keys to entities

3.16
key encryption key
cryptographic key that is used for the encryption or decryption of other keys

[ISO/IEC 19790:2006]

3.17
leaf node
node in a tree which is not a parent of any other node, i.e. has no child nodes

3.18
logical key hierarchy
tree used for managing the shared secret key and key encryption keys

3.19
logical key structure
logical structure to manage keys

NOTE This structure has no correlation with the network topology.

3.20
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

[ISO/IEC 11770-3:2008]

3.21
one-way function with trapdoor
function that is known to be easy to compute but hard to invert unless some secret information (trapdoor) is known

3.22
parent node of node *c*
node on which node *c* hangs

3.23
perfect backward secrecy
security condition in which a joining entity cannot obtain any former shared secret keys

3.24
perfect forward secrecy
security condition in which a leaving entity cannot obtain any subsequent shared secret keys

3.25
random number
time variant parameter whose value is unpredictable

[ISO/IEC 11770-1:2010]

3.26
rekeying
process of updating and redistributing the shared secret key, and optionally, key encryption keys

NOTE This process is executed by the key distribution centre.

3.27
root node
node in a tree which is not a child of any other node

3.28
shared secret key
key which is shared with all the active entities via a key establishment mechanism for multiple entities

3.29
symmetric key based key establishment mechanism for multiple entities
process of establishing a shared secret key between all active entities, using symmetric cryptographic techniques

3.30
tree
connected, acyclic graph with an identified special vertex, the root node

4 Symbols and abbreviations

AK	Ancestor key
BWK_i	Backward key for the time instance *i*
CK	Child key
COM(*X,Y*)	Function, which generates from the data items *X* and *Y* a key designed to be applied as key of the used encryption algorithm
CUT(*k*,*S*)	Function which outputs a substring of length *k* of the least significant bits of a string *S* of bits
d	Number of children of a parent node (see term *d*-ary tree)
e(*K*,*Z*)	Result of encrypting data *Z* with a symmetric encryption algorithm using the secret key *K*
f	One-way function with trap door
f^{-1}	Inverted function of *f,* which requires the trapdoor of *f*
FWK_i	Forward key for the time instance *i*
g_1	One-way function
g_2	One-way function
h	Number of ancestor nodes of a leaf node excluding the root node
IK	Individual key
IK x	Individual key shared between entity *x* and the key distribution centre
KDC	Key distribution centre
KEK	Key encryption key
LKH	Logical key hierarchy
m	Number of entities connected to the hub in a star structure
MAC(*K*,*Z*)	MAC function as defined in ISO/IEC 9797 using key *K* and data *Z*
$r_{BWKinit}$	Random number to initialize the backward key chain

$r_{FWKinit}$	Random number to initialize the forward key chain
RSA	Digital signature mechanism as defined in ISO/IEC 14888-2
s	Private key
SHA-1	Dedicated hash function as defined in ISO/IEC 10118-3
SSK	Shared secret key
v	Public key
$X\|\|Y$	Result of concatenating data items X and Y in that order

5 Requirements

The key establishment mechanisms specified in this part of ISO/IEC 11770 realize point-to-multipoint key communication by using logical key structures. The point-to-multipoint communication requires a key updating process when a new entity joins or an entity leaves the communication in order to maintain the secrecy of the communication.

a) There are two types of security requirements, perfect backward secrecy and forward secrecy and backward secrecy and forward secrecy with intervals. One of these security requirements shall be chosen depending on the security requirements of the particular application. Key establishment mechanisms for multiple entities require two different rekeying methods according to the security requirements: individual rekeying and batched rekeying. Individual re-keying provides perfect backward secrecy and forward secrecy, and batched rekeying provides backward secrecy and forward secrecy with interval T. The rekeying method and parameter setting have a strong influence on the security requirements; thus, they shall be determined according to the security policy of the application.

b) The encryption algorithm shall be chosen in accordance with the following:

   1) A symmetric encryption algorithm shall be chosen from among those standardised in ISO/IEC 18033-3 and ISO/IEC 18033-4.

   2) If a block cipher encryption algorithm is used, then the Mode of Operation employed shall be one of those standardised in ISO/IEC 10116, ISO/IEC 18033-3, ISO/IEC 18033-4 and ISO/IEC 19772. An encryption algorithm used for key encryption shall provide integrity, and input length shall be more than 128 bits. One of the mechanisms in ISO/IEC 19772 shall be used for integrity protection.

c) The shared secret key is established using either a secure or insecure communication channel. At least the individual key shall be exchanged between the key distribution centre and each entity using a secure channel in order to allow secure communication. A secure communication channel is one where an attacker cannot eavesdrop or tamper with messages in the channel.

d) The key establishment mechanisms in this part of ISO/IEC 11770 require the use of random numbers to generate the shared secret key, and optionally, key encryption keys. For means of generating random numbers, see ISO/IEC 18031.

6 Tree based key establishment mechanisms for multiple entities

6.1 General model

Key establishment for multiple entities enables the transmission of a message to all the entities, such that any active entities can decrypt the message correctly and any coalition of inactive entities cannot decrypt it. All the active entities share the shared secret key that is used to encrypt the message. An active entity may dynamically change to being inactive, and vice versa. The key distribution centre updates the shared secret key to prevent the joining entity from obtaining the former messages and the leaving entity from obtaining the subsequent messages.

Figure 1 shows the general model of key establishment for multiple entities, in which the key distribution centre can communicate with all the entities. The communication between the key distribution centre and entities does not need to be secure. The key distribution centre and each entity shall share a distinct individual key. The key distribution centre is responsible for distributing the shared secret key to all the active entities. The join/leave request is represented by (1) and the distribution of keys to the entities by (2), (3), ..., (n+1). From ii onward, the order in which the updates take place is not important.

NOTE if one of the entities that knows the shared secret key cannot be contacted for a period of time, that entity may miss a key update message, and cannot compute the updated shared secret key.

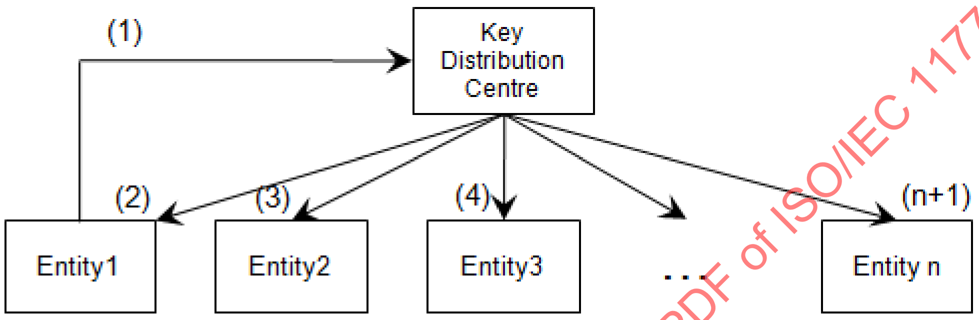

Figure 1 — General model of key establishment for multiple entities

6.2 Joining process

An entity sends a joining request to the key distribution centre in order to start obtaining the shared secret key. The key distribution centre executes the rekeying process after the requesting entity was accepted to join in the case where individual rekeying is adopted. On the other hand, the key distribution centre does not execute the rekeying process in the joining process in the case where batched rekeying is adopted.

6.3 Leaving process

An entity sends a leave request to the key distribution centre in order to stop obtaining the shared secret key. The key distribution centre executes rekeying after the leaving entity has left in the case where individual rekeying is adopted. On the other hand, there is no explicit leaving process in the case where batched rekeying is adopted. However, the key distribution centre shall record the leaving entities for the next rekeying process.

NOTE When the batched rekeying is used, the entity leaving the group can still decrypt communications in the group until the next batch rekeying takes place.

6.4 Rekeying process

The key distribution centre updates the shared secret key, and optionally, key encryption keys in order to satisfy security requirements. This process is executed in both the joining and leaving processes in the case where individual rekeying is adopted, and executed at regular time intervals in the case where batched rekeying is adopted.

6.5 Logical key structure

6.5.1 Star based structure

Key establishment mechanisms for multiple entities can be classified by their logical structures and are used to assign keys to entities. Figure 2 shows the star-based structure.

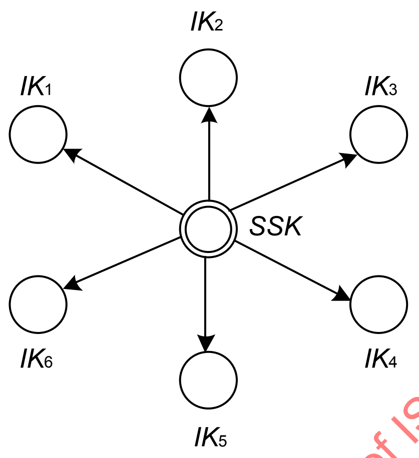

Figure 2 — Star-based structure

6.5.2 *d*-ary tree based structure

Figure 3 shows the binary tree structure where $d = 2$. A shared secret key is assigned to the root node of the tree. Each individual key is assigned to the leaf nodes of the tree. Additionally, key encryption keys are assigned to the other nodes. The key encryption keys are shared by multiple entities whose individual keys are assigned to the descendant of the node to which the key encryption key is assigned. The communication cost of the leaving process can be reduced by using key encryption keys. Each entity has all the keys assigned to the nodes on the path from the root node to the leaf node, to which the individual key of the entity is assigned. Thus, the number of keys an entity has is proportional to the logarithm of the total number of entities.

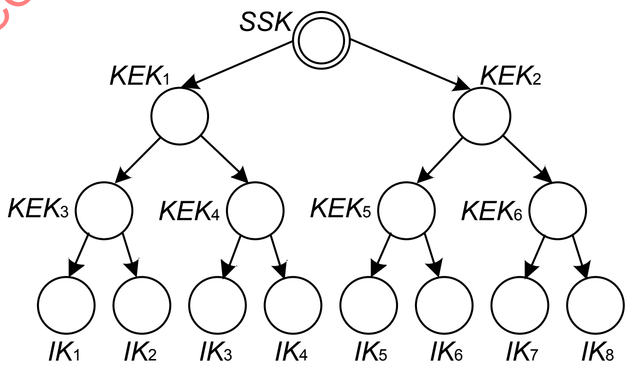

Figure 3 — *d*-ary tree based structure

6.5.3 General tree based structure

A general tree based structure can be used as the logical key structure. The general tree based structure makes use of a *d*-ary tree based structure where *m* entities construct a cluster. This structure can be considered as a hybrid of the star-based structure with *m* clients and the *d*-ary tree based structure.

This structure can be used to optimize the efficiency of key establishment mechanisms (see Annex B). Figure 4 shows the tree based structure where d = 2 and m = 4. The general tree based structure contains a d-ary tree based structure, however, the opposite does not hold. For example, the tree based structure in Figure 4 is not a d-ary tree based structure.

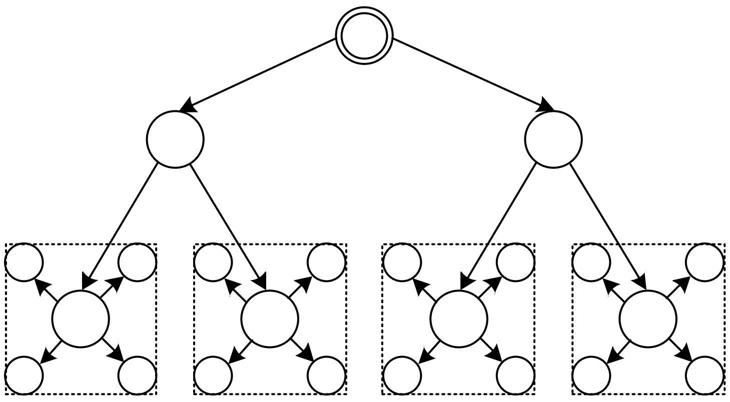

Figure 4 — General-tree based structure

6.6 Symmetric key based key establishment mechanisms

6.6.1 Mechanism 1 - Key establishment mechanism with individual rekeying

This document defines symmetric-key based key establishment mechanisms based on general tree based structure: 1) mechanism with individual rekeying and 2) mechanism with batched rekeying. In the mechanism with individual rekeying, the rekeying process is executed whenever an entity joins or leaves.

This mechanism is based on a tree based structure with individual rekeying.

a) Joining process

It is assumed that there is a set of n entities $\{u_1, u_2, ..., u_n\}$, and the entity u_{n+1} joins. Let $AK(l,u_i)$ be the ancestor key of entity u_i that is assigned to the l-th layer from the root node of the logical key hierarchy. h denotes the height of the logical key hierarchy.

1) The entity u_{n+1} sends a join request to the key distribution centre.

2) The key distribution centre assigns the individual key of u_{n+1} (i.e., $IK\ u_{n+1}$) to a leaf node of the logical key hierarchy.

3) The key distribution centre generates random numbers and updates the ancestor keys of the individual key of u_{n+1} using these numbers. SSK, $AK(1,u_{n+1})$, $AK(2,u_{n+1})$, ..., $AK(h,u_{n+1})$ are updated to SSK', $AK'(1,u_{n+1})$, $AK'(2,u_{n+1})$, ..., $AK'(h,u_{n+1})$, respectively.

4) The key distribution centre encrypts each updated key with the old key, and broadcasts it. That is, $e(SSK,SSK')$, $e(AK(1,u_{n+1}),AK'(1,u_{n+1}))$, $e(AK(2,u_{n+1}),AK'(2,u_{n+1}))$, ..., and $e(AK(h,u_{n+1}),AK'(h,u_{n+1}))$ are broadcast.

5) Each entity obtains the updated keys using the old keys.

6) The key distribution centre encrypts the updated keys $SSK'||AK'(1,u_{n+1})||AK'(2,u_{n+1})||...||AK'(h,u_{n+1})$ by the individual key of u_{n+1}, and sends $e(IK\ u_{n+1},SSK'||AK'(1,u_{n+1})||AK'(2,u_{n+1})||...||AK'(h,u_{n+1}))$ to u_{n+1}.

7) The entity u_{n+1} obtains the keys.

b) Leaving process

It is assumed that there are n entities $\{u_1, u_2, ..., u_n\}$, and the entity u_j ($1 \leq j \leq n$) leaves. Let $AK(l,u_i)$ be the ancestor key of entity u_i that is assigned to the l-th layer from the root node of the logical key hierarchy. h denotes the height of the logical key hierarchy.

1) The key distribution centre generates random numbers and updates the ancestor keys of the individual key of u_j using these numbers. SSK, $AK(1,u_j)$, $AK(2,u_j)$, ..., $AK(h,u_j)$ are updated to SSK', $AK'(1,u_j)$, $AK'(2,u_j)$, ..., $AK'(h,u_j)$, respectively.

2) The key distribution centre encrypts each updated key with its child keys except the individual key of u_j and broadcasts them. For example, the SSK' is encrypted with the child keys CK_1, CK_2, ..., CK_d, and $e(CK_1,SSK')$, $e(CK_2,SSK')$, ..., and $e(CK_d,SSK')$ are broadcast.

NOTE In the case that child keys have been updated, the updated child keys are used.

3) Each entity obtains the updated keys using the child keys.

An example of mechanism 1:

1) Phase 1

An example of the joining process of mechanism 1 is illustrated in Figure 5. It is assumed that the key distribution centre uses the logical key hierarchy in Figure 6 and the entity H is joining.

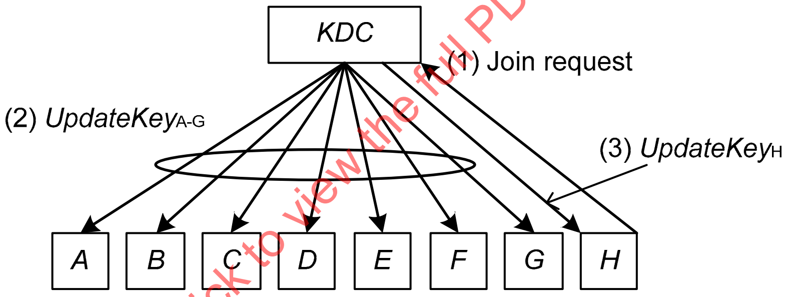

Figure 5 — Joining process of mechanism 1 — mechanism with individual rekeying

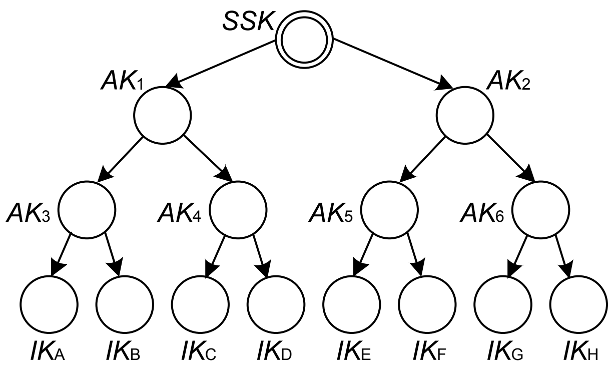

Figure 6 — Logical key procedure

The updated keys ($UpdatedKey_{A\text{-}G}$), broadcasted by the key distribution centre to A, B, C, D, E, F, and G is:

$UpdatedKey_{A\text{-}G} = e(SSK,SSK')||e(AK_2,AK'_2)||e(AK_6,AK'_6)$.

The updated keys ($UpdatedKey_H$), sent by the key distribution centre to H is:

$UpdatedKey_H = e(IK_H, SSK'||AK'_2||AK'_6)$.

(1) *H* sends Join request to key distribution centre.

(2) The key distribution centre generates and broadcasts $UpdatedKey_{A\text{-}G}$ *to A, B, C, D, E, F,* and *G*.

(3) The key distribution centre generates and sends $UpdateKey_H$ to *H*.

NOTE The key distribution centre and H share the individual key of H in advance.

2) Phase 2

An example of the leaving process of mechanism 1 is illustrated in Figure 7. It is assumed that the key distribution centre uses the logical key hierarchy in Figure 6 and the entity *H* is leaving.

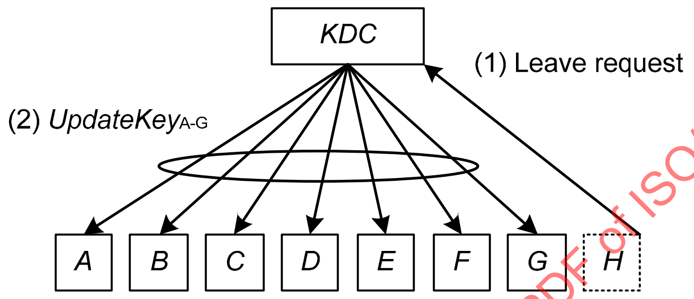

Figure 7— Leaving process of mechanism 1 — mechanism with individual rekeying

The form of the updated keys ($UpdatedKey_{A\text{-}G}$), broadcast by the key distribution centre to *A*, *B*, *C*, *D*, *E*, *F*, and *G* is:

$UpdatedKey_{A\text{-}G} = e(AK_1, SSK')||e(AK'_2, SSK')||e(AK_5, AK'_2)||\ e(AK'_6, AK'_2)||\ e(IK_G, AK'_6)$.

(1) *H* sends Leaving request to key distribution centre.

(2) The key distribution centre generates and broadcasts $UpdatedKey_{A\text{-}G}$ to *A*, *B*, *C*, *D*, *E*, *F*, and *G*.

6.6.2 Mechanism 2 - Key establishment mechanism with batched rekeying

In the mechanism with batched rekeying, the rekeying process is periodically executed.

a) Joining process

It is assumed that there are *n* entities $\{u_1, u_2, ..., u_n\}$, and the entity u_{n+1} joins. Let $AK(l,u_i)$ be the ancestor key of entity u_i that is assigned to the *l*-th layer from the root node of the logical key hierarchy. *h* denotes the height of the logical key hierarchy.

1) The entity u_{n+1} sends a join request to the key distribution centre.

2) The key distribution centre assigns the individual key of u_{n+1} to a leaf node of the logical key hierarchy.

3) The key distribution centre encrypts the ancestor keys of the individual key of u_{n+1} by the individual key of u_{n+1}. Then, the key distribution centre sends $e(IK\ u_{n+1}, SSK||AK(1, u_{n+1})||AK(2, u_{n+1})||\ ...||AK(h, u_{n+1}))$ to u_{n+1}.

4) The entity u_{n+1} obtains *SSK* and $AK(1, u_{n+1}), AK(2, u_{n+1}), \ldots, AK(h, u_{n+1})$.

b) Leaving process

The leaving entity sends Leaving request to key distribution centre.

NOTE Rekeying is not executed in the leaving process of the mechanism with batched rekeying.

c) Rekeying process

This process executed at regular time intervals. It is assumed that there is a set of n entities $\{u_1, u_2, ..., u_n\}$ and that the entities of the set $\{u_{i1}, u_{i2}, .. u_{ik}\}$ left during a rekeying interval. k is the number of leaving entities during a rekeying interval i. Let $AK(l,u_i)$ be the ancestor key of entity u_i that is assigned to the l-th layer from the root node of the logical key hierarchy. h denotes the height of the logical key hierarchy.

1) The key distribution centre generates random numbers and updates the ancestor keys of the individual key of $u_{i1}, u_{i2}, ..., u_{ik}$ using the random numbers. SSK, $AK(1,u_{i1})$, $AK(1,u_{i2})$, …, $AK(1,u_{ik})$, $AK(2,u_{i1})$, $AK(2,u_{i2})$, …, $AK(2,u_{ik})$, $AK(h,u_{i1})$, $AK(h,u_{i2})$, …, $AK(h,u_{ik})$ are updated to SSK', $AK'(1,u_{i1})$, $AK'(1,u_{i2})$, …, $AK'(1,u_{ik})$, $AK'(2,u_{i1})$, $AK'(2,u_{i2})$, …, $AK'(2,u_{ik})$, $AK'(h,u_{i1})$, $AK'(h,u_{i2})$, …, $AK'(h,u_{ik})$, respectively.

2) The key distribution centre encrypts each updated key with its child keys except the individual key of u_{i1}, u_{i2}, …, u_{ik} and broadcasts them. For example, the SSK' is encrypted with the child keys CK_1, CK_2, ..., CK_d, and $e(CK_1,SSK')$, $e(CK_2,SSK')$, ..., and $e(CK_d,SSK')$ are broadcast.

NOTE In the case that child keys have been updated, the updated child keys are used.

3) Each entity obtains the updated keys using the child keys.

An example of mechanism 2:

1) Phase 1

An example of the joining process of mechanism 2 is illustrated in Figure 8. It is assumed that the key distribution centre uses the logical key hierarchy in Figure 6 and the entity H is joining.

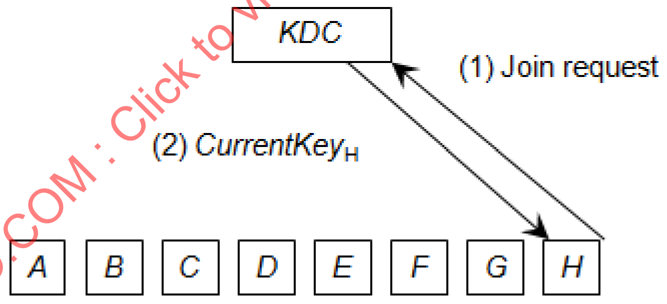

Figure 8 — Joining process of mechanism 2 — mechanism with batched rekeying

The form of the updated keys ($CurrentKey_H$), sent by the key distribution centre to H is:

$CurrentKey_H = e(IK_H, SSK||AK_2||AK_6)$.

(1) H sends Join request to key distribution centre.

(2) The key distribution centre generates and sends $CurrentKey_H$ to H.

2) Phase 2

An example of the leaving process of mechanism 2 is illustrated in Figure 9. It is assumed that the key distribution centre uses the logical key hierarchy in Figure 6 and the entities A and H have left.

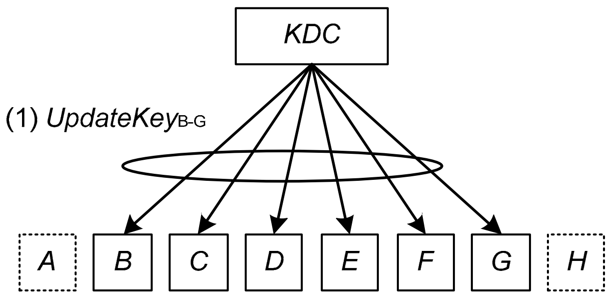

Figure 9 — Rekeying process of mechanism 2 — mechanism with batched rekeying

The updated keys ($UpdatedKey_{B\text{-}G}$), broadcasted by the key distribution centre to B, C, D, E, F, and G is:

$UpdatedKey_{B\text{-}G} = e(AK'_1,SSK')||e(AK'_2,SSK')||e(AK'_3,AK'_1)||\ e(AK_4,AK'_1)||$

$e(AK_5,AK'_2)||e(AK'_6,AK'_2)||e(IK_B,AK'_3)||e(IK_G,AK'_6)$.

The key distribution centre generates and broadcasts $UpdatedKey_{B\text{-}G}$ to B, C, D, E, F, and G.

7 Key chain based group key management

In order to limit the validity period of keys, key chains are useful. Key chains can limit access to encrypted information in future sessions, past sessions or both. In a key chain the individual members are dependent on each other. Using a start value a first key is calculated. The second key is calculated to be dependent on the first one, the third key calculated to be dependent on the second key, etc. If the decentralized entity is able to perform the calculation, it allows for efficient decentralized key generation. Key chain length may be limited to a predetermined number of keys, or not limited at all. Any particular instance of a key in the key chain, and its time interval for which it is valid, determines the time interval in which the encrypted information can be decrypted using that particular key.

In this standard two types of key chains are considered. The first type of key chain limits access to encrypted information in future (forward key chains), whereas the second type of key chain limits access to encrypted information in the past (backward key chains).

The calculation of keys within the chains is based on one-way functions or one-way functions with trapdoor. One-way functions allow calculation in one direction only. This means that starting with the result of a one-way function, calculation of the start value cannot be performed. One-way functions with trapdoor allow the use in both directions. In one direction it is easy to calculate the result of a start value, but in order to get the start value from a given result a secret is needed to use the trapdoor functionality.

The case, that a one-way function with trapdoor is used for the forward key chain and a one-way function is used for the backward key chain, will be considered in 8. The use of a one-way function with trapdoor for the forward key chain will result in an unlimited forward key chain.

The case, that one-way functions are used for the forward key chain as well as for the backward key chain, will be considered in 9. The use of a one-way function instead of a one-way function with trapdoor for the forward key chain will result in a limited forward chain limited by the start value.

Special measures have to be implemented to prevent collusion attacks. If one entity has access to one part of a key chain, and another entity has access to another part of the same key chain, they can reconstruct the key chain from the lowest start of both parts of the key chain till the highest end of both parts of the key chain. The keys of a key chain have to be handled like secret keys, whose forwarding to other entities has to be prevented.

8 Key chain based group key management with unlimited forward key chain

8.1 Calculations by the key distribution centre

8.1.1 Key chains

The key distribution entity needs to set up the key chains prior to normal operation. This means, that the forward and backward key chains have to be defined. The forward chain would then be

$$FWK = \{ FWK_0, FWK_1, \ldots, FWK_{n-1}, FWK_n, \ldots \}$$

and for the backward chain

$$BWK = \{ BWK_0, BWK_1, \ldots, BWK_{n-1}, BWK_n, \ldots \}$$

where all parties can continue the backward key chain indefinitely, and the key distribution centre can continue the forward key chain indefinitely by use of the trapdoor.

Depending on the functions used for the calculation of *FWK* and *BWK* the start values FWK_0 and BWK_0 have to be chosen.

8.1.2 Forward secrecy

Unlimited forward secrecy can be realized by defining an appropriate forward key chain. Any of the keys contained in the chain can be evaluated by

$$FWK_i = f^{-1}(FWK_{i-1}),\ i = 1,2,3,\ldots$$

where f^{-1} uses the trapdoor of *f*. The iterative relation can be written as

$$FWK_T = (f^{-1})^T(FWK_0) := \underbrace{f^{-1}(\, f^{-1}(\ldots f^{-1}(FWK_0)))}_{T\ times},\quad T = 1,2,3,\ldots$$

The key distribution centre is able to continuously calculate the forward key chain FWK_0, … FWK_T,… as outlined in Figure 10. This calculation can only be done by the key distribution centre since no user entity is in possession of the trapdoor.

With this chain no access to encrypted information encrypted by FWK_{T+i}, i > 0 for the user entities, if they possess FWK_T.

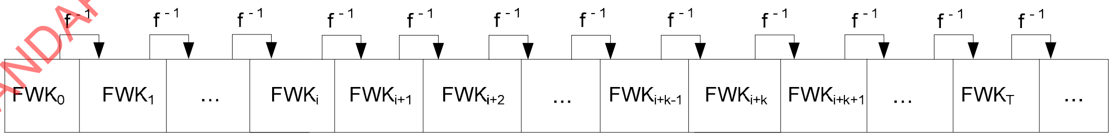

Figure 10 — Construction of a forward key chain using a one-way function with trapdoor

The following is an example of the forward key chain construction of the key distribution centre using RSA as one-way function with trapdoor (*n* is the modulus, *v* the public key and *s* the private key):

1) Initially the key distribution centre generates a secure random number $r_{FWKinit}$ as the start value. The key distribution centre uses the digital signature ability of RSA. Therefore the private key (or trapdoor) is used for signing the start value. This corresponding signature is the first key FWK_0. That is,

$FWK_0 = r_{FWKinit}{}^{s} \bmod n$ is calculated

2) The next element of the chain is constructed by giving the RSA signature function FWK_0 as input and using the output of the function as FWK_1: $FWK_1 = FWK_0^s \bmod n$.
New keys are generated the same way. The input is always the current last key of the chain FWK_i and the output is FWK_{i+1}: $FWK_{i+1} = FWK_i^s \bmod n$.

3) If required, the key distribution centre can always extend the chain by calculating new signatures based on the current last key.

The use of the RSA signature function and the generation of the start value are based on the requirements given by ISO/IEC 14888-2.

8.1.3 Backward secrecy

Unlimited backward secrecy can be realized by defining a backward key chain. Any of the keys contained in the chain can be evaluated by

$$BWK_{i+1} = g_1(BWK_i),\ i = 0,1,2,\dots$$

where g_1 represents a one-way function. The iterative relation can be written as

$$BWK_T = g_1^{T}(BWK_0) := \underbrace{g_1(g_1(\dots g_1(BWK_0)))}_{T\ times},\ T = 1,2,3,\dots$$

The key distribution centre is able to continuously calculate the backward key chain $BWK_0 \dots BWK_T \dots$ as it is outlined in Figure 11.

This chain does not allow access to encrypted information for $t < T$ in the case that the user entity knows BWK_T. Any key chain $BWK_i \dots BWK_T, \dots$ gives restriction for any time $t < t_i$.

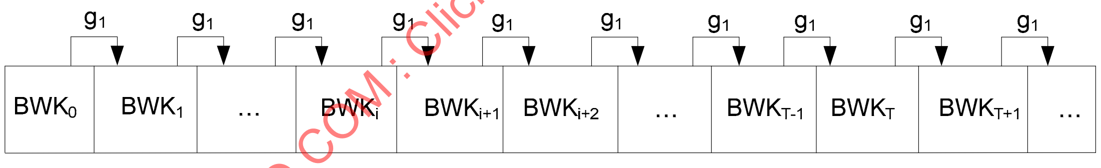

Figure 11 — Construction of a backward key chain using a one-way function

The following is an example of the backward key chain construction of the key distribution centre using the hash-function SHA-1 as one-way function:

1) key distribution centre generates a secure random number $r_{BWKinit}$ as a start value.

2) The key distribution centre uses SHA-1 for hashing the random start value to a fixed size of 160 Bit. The output is used as the first key BWK_0. That is, BWK_0 = SHA-1($r_{BWKinit}$) is calculated.

3) The next element of the chain is constructed using BWK_0 as input for SHA-1 and using the output of the function as BWK_1: BWK_1 = SHA-1(BWK_0).

4) New keys of the chain are generated using SHA-1. The input is always the current last key of the chain BWK_i and the output is BWK_{i+1}: BWK_{i+1} = SHA-1(BWK_i).

The use of the SHA-1 hash function and the generation of the start value are based on the requirements given by ISO/IEC 10118-3.

8.1.4 Forward and backward secrecy

For combined forward and backward secrecy, the individual key chains are connected to allow restricted access in both directions. This is realized by another function COM used to combine the individual keys FWK_i and BWK_i. The function *COM* may depend on a key *K* shared between the key distribution centre and the user entities. The combination results in key

$$K_i = COM(FWK_i, BWK_i),\ i = 1,2,\ldots$$

It shall be checked whether any resulting K_i satisfies the security requirements of the encryption algorithm, which will use K_i as secret key. Even if it is very unlikely to produce weak keys, counter-measures have to be created and most likely a rekeying is needed in such a case.

Example 1 for the case, that K_i does not depend on a key *K*:

$$COM = CUT(k, \text{SHA-1}(FWK_i \parallel BWK_i))$$

The length *k* of the output of *CUT* is the required length of K_i .

Following steps are performed:

1) FWK_i and BWK_i are concatenated

2) The hash value of (FWK_i || BWK_i) is calculated by SHA-1

3) The substring of the *k* least significant bits of the hash value is chosen.

4) The result is used as the key K_i for deciphering of the encrypted data.

Example 2 for the case, that K_i does depend on a key *K*:

$$COM = CUT(k, MAC(K, FWK_i \parallel BWK_i))$$

Such a *MAC* function shall be chosen from ISO/IEC 9797, that the length of the *MAC* output is not smaller than *k*.

Following steps are performed:

1) FWK_i and BWK_i are concatenated

2) The MAC value of (FWK_i || BWK_i) is calculated by using a *MAC* function of ISO/IEC 9797

3) The substring of the *k* least significant bits of the *MAC* is chosen

4) The result is used as the key K_i for deciphering of the encrypted data.

8.2 Calculations by the client entity

8.2.1 Key chains

In order to access the encrypted information, the receiver has to have the same keys as the key distribution centre during the period. It is not efficient to transmit the whole key chain from key distribution centre to the receiver; due to this the receiver will be enabled to calculate the key chain(s). To keep the restriction for access alive, the receiver shall neither be in possession of the trapdoor function nor able to use the trapdoor of the one-way function nor in possession of the start value of the backward key chain.

In general, the receiver will be given values FWK_{i+k} for the forward key chain and BWK_i for the backward key chain. With this, no access for $t < t_i$ or $t > t_{i+k}$ is possible.

The receiver shall also be given the functions necessary to calculate the key chains. For the forward chain f is needed and for the backward chain g_1 will be given.

8.2.2 Forward secrecy

The receiver constructs the forward key chain using the given value FWK_{i+k}. The mathematical relation is

$$FWK_{i+k-j-1} = f(FWK_{i+k-j}),\ j = 0,1,2,\ldots,i+k-1$$

whereas the iterative relation can be written as

$$FWK_j = f^{\,i+k-j}(FWK_{i+k}) := \underbrace{f(f(\ldots f(}_{(i+k-j)\text{ times}}FWK_{i+k}))),\ j = 0,\ldots,i+k-1$$

The resulting key chain written as a set for an interval T is

$$FWK = \{FWK_{i+k-T}, FWK_{i+k-T+1}, \ldots, FWK_{i+k-1}, FWK_{i+k}\}$$

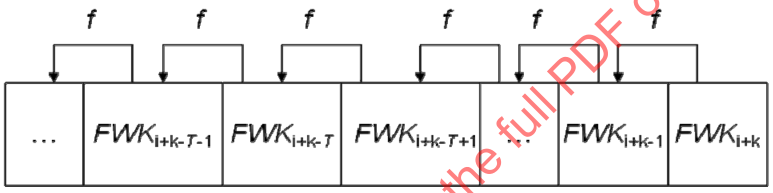

Figure 12 — Calculation of a forward key chain

The following is an example of the forward key calculation of the client entity correspondingly to the example of 8.1.2:

1) To calculate the forward key chain as outlined in Figure 12, the client entity shall be in possession of his personal start value FWK_{i+k} and the public key for the RSA signature verification. That is, the client entity shall know n and v. These values shall have been previously received from the key distribution centre.

2) The next key of the chain is calculated by using the RSA signature verification. That is, the client entity calculates $FWK_{i+k-1} = FWK^{v}_{i+k}$ mod n.

3) The following keys of the chain are calculated the same way. The client entity calculates the RSA verification function with the current last key of the forward key chain as the input, until the desired key has been calculated. That is, $FWK_{i+k-T} = FWK_{i+k-T+1}{}^{v}$ mod n has to be calculated.

8.2.3 Backward secrecy

The receiver constructs the backward key chain using the given value BWK_i. The mathematical relation is

$$BWK_{i+j+1} = g_1(BWK_{i+j}),\ \ j = 0,1,2,\ldots$$

whereas the iterative relation can be written as

$$BWK_{i+T} = g_1{}^{T}(BWK_i) := \underbrace{g_1(g_1(\ldots g_1(}_{T\ times}BWK_i)))$$

The resulting key chain written as a set for an interval T is

$$BWK = \{BWK_i, BWK_{i+1}, \ldots, BWK_{i+T-1}, BWK_{i+T}\}$$

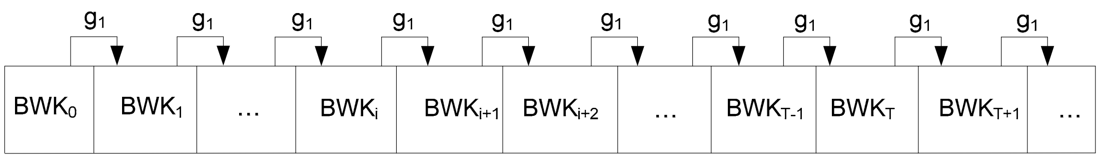

Figure 13 — Calculation of a backward key chain

The following is an example of the backward key calculation of the client entity correspondingly to the example of 8.1.3:

1) To calculate the backward key chain as outlined in Figure 13, the client entity shall be in possession of the personal start value BWK_i which shall previously be received from the key distribution centre.

2) The client entity uses SHA-1 for hashing the start value in order to get the next key of the chain. That is, BWK_{i+1} = SHA-1(BWK_i) is calculated.

3) New keys of the chain are generated using SHA-1. The input is always the current last key of the chain BWK_{i+k} and the output is BWK_{i+k+1} = SHA-1(BWK_{i+k}) as shown in Figure 13.

8.2.4 Forward and backward secrecy

For combined forward and backward secrecy, the individual key chains are connected to allow restricted access in both directions. Analogous to 8.1.4 this is realized by another function COM used to combine the individual keys FWK_i and BWK_i. The function *COM* may depend on a key *K* shared between the key distribution centre and the user entities. The combination results in key

$$K_i = COM(FWK_i, BWK_i),\ i = 1,2,\ldots$$

Examples for the function *COM* are given in 8.1.4

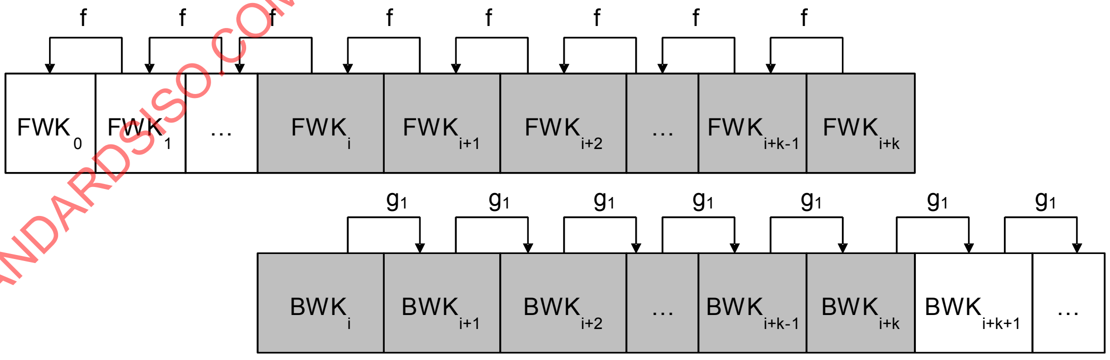

Figure 14 — Access limitation by combination of the key chains

The shaded key chain values in Figure 14 show the valid keys for the client entity.

9 Key chain based group key management with limited forward key chain

9.1 Calculations by the key distribution centre

9.1.1 Key chains

In this case for both key chains one-way functions are used. The key distribution entity needs to set up the key chains prior to normal operation. This means that, depending on what is needed, the forward and/or backward key chain has/have to be defined. The result for the forward chain would then look like

$$FWK = \{FWK_0, FWK_1, \ldots, FWK_{n-1}, FWK_n\}$$

and for the backward chain

$$BWK = \{BWK_0, BWK_1, \ldots, BWK_{n-1}, BWK_n, \ldots\}$$

Both key chains are calculated by usage of one-way functions.

In this case the start values are FWK_n and BWK_0. The index n of the start value FWK_n has to be chosen appropriately based on the application and implementation aspects, because it means the end of the forward key chain.

9.1.2 Forward secrecy

The key distribution centre constructs the forward key chain using the given start value FWK_n. Any of the keys contained in the limited chain can be evaluated by

$$FWK_{n-i-1} = g_2(FWK_{n-i}),\ i = 0,1,2,\ldots,n-1$$

where g_2 represents a one-way function. The iterative relation can be written as

$$FWK_{n-i} = g_2(FWK_n) := \underbrace{g_2(g_2(\ldots g_2}_{\text{i times}}(FWK_n))),\ i = 1,2,3,\ldots,n$$

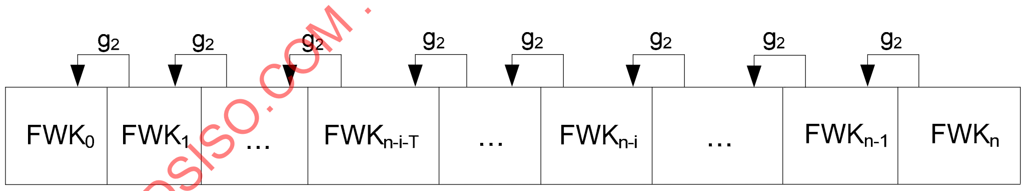

Figure 15 — Construction of a forward key chain by use of a one-way Function

The following is a description of the forward key chain construction of the key distribution centre using the function SHA-1 as one-way function:

1) key distribution centre generates a secure random number as a start value $r_{FWKinit}$.

2) The key distribution centre uses SHA-1 for hashing the random start value to a fixed size of 160 Bit. The output is used as the first key FWK_n. That is, FWK_n = SHA-1($r_{FWKinit}$) is calculated.

3) The next element of the chain is constructed using FWK_n as input for SHA-1 and using the output of the function as FWK_{n-1}: FWK_{n-1} = SHA-1(FWK_n).

4) New keys of the chain are generated using SHA-1. The input is always the current last key of the chain FWK_{n-i+1} and the output is FWK_{n-i}: FWK_{n-i} = SHA-1(FWK_{n-i+1}).

5) The calculation finishes as soon as the key distribution centre declares one final value FWK_0.