|  |  |
| --- | --- |
| Project | **IEEE 802.21 MIHS****<**[**http://www.ieee802.org/21/**](http://www.ieee802.org/21/)**>** |
| Title | **Suggested remedy for SB Comment i-12** |
| DCN | **21-14-0174-00-MuGM** |
| Date Submitted | **November 5, 2014** |
| Source(s) | Yoshihiro Ohba and Yoshikazu Hanatani (Toshiba) |  |
| Re: | IEEE 802.21d Sponsor Ballot comment resolution |
| Abstract | This document describes the detail of DCN 0173-01 to propose remedy for SB comment i-12 about usage of Bloom Filter. |
| Purpose | For Sponsor Ballot Comment Resolution |
| Notice | This document has been prepared to assist the IEEE 802.21 Working Group. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein. |
| Release | The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE’s name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE’s sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that IEEE 802.21 may make this contribution public. |
| Patent Policy | The contributor is familiar with IEEE patent policy, as stated in [Section 6 of the IEEE-SA Standards Board bylaws](http://standards.ieee.org/guides/opman/sect6.html#6.3) <[http://standards.ieee.org/guides/bylaws/sect6-7.html#6](http://127.0.0.1:4664/cache?event_id=757737&schema_id=1&s=5X0vID10lu_E6yrIkWkNd4Wz2H8&q=hancock)> and in *Understanding Patent Issues During IEEE Standards Development* <http://standards.ieee.org/board/pat/faq.pdf> |

# Comments

# Comment i-12

When multiple certificates are revoked, it should be able to revoke them using a single message. In that case, the size of the message should be reduced as much as possible.

# Suggested remedy for Comment i-12

Define a list or Bloom Filter of revoked certificates. A technical contribution will be submitted for further details.

# Proposed resolution

We can use Bloom Filter to reduce the length of CompleteSubtree TLV.

In some use case, the length of Group Manipulation Command shall be reduced as much as possible. So, we propose CompleteSubtree TLV to be “Optional”.

* + If GroupKeyData TLV is attached, CompleteSubtree TLV can be omitted.
	+ If GroupKeyData TLV is not attached, CompleteSubtree TLV shall be attached.

Revise following subclauses.

* + 4. Abbreviations and acronyms (From D/06)
	+ 7.4.31.3 MIH\_MN\_Group\_Manipulate.response (From D/06)
	+ 7.4.32.1 MIH\_Net\_Group\_Manipulate.request (From DCN 167-01)
	+ 8.6.1.24 MIH\_MN\_Group\_Manipulate response (From D/06)
	+ 8.6.1.25 MIH\_Net\_Group\_Manipulate request (From DCN 167-01)
	+ 8.6.1.26 MIH\_Net\_Group\_Manipulate indication (From DCN 167-01)
	+ 9.5.2.1 Master Group Key Wrapping (From D/06)
	+ 9.5.2.2 Master group key unwrapping (From D/06)
	+ 9.5.3.1.1 Group Manager (From D/06)
	+ 9.5.3.1.2 MIHF of a PoS with Group Manager (From DCN 167-01)
	+ 9.5.3.1.3 Receiving procedure for group manipulation commands (From DCN 167-01)
	+ 9.6.1 Group session key derivation (From D/06)
	+ 9.6.5 Group key distribution Ciphersuites (From D/06)
	+ Table F.24- Data type for security (From D/06)
	+ Table L.2- Type values for TLV encoding (From D/06)
* Revise 4 Abbreviations and acronyms
	+ Add following abbreviation.

MIGKVK Media Independent Group Key Verification Key

* Revise 7.4.31.3 MIH\_MN\_Group\_Manipulate.response

MIH\_MN\_Group\_Manipulate.response (

DestinationIdentifier,

TargetIdentifier,

TransportAddress,

MasterGroupKey,

SubgroupRange,

UserSpecificData,

CompleteSubtree,

ComplementSubtreeFlag,

GroupKeyData,

VerifyGroupKey,

GroupStatus

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies the MIHF ID of the destination of the primitive. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| TransportAddress | TRANSPORT\_ADDR | (Optional) Multicast or unicast address corresponding with the target group identifier. A unicast address may be used for a two-member group. |
| MasterGroupKey | MGK | (Optional) The master group key associated with the target MIHF group identifier. |
| SubgroupRange**a** | SUBGROUP\_RANGE | (Optional) Subgroup to process the command. |
| UserSpecificDatab | OCTET\_STRING | (Optional) Auxiliary data. |
| CompleteSubtreec | COMPLETE\_SUBTREE | (Optional) Complete Subtree data. |
| ComplementSubtreeFlagd | SUBTREE\_FLAG | (Optional) Flag to interpret the complete subtree data (See 9.5.2) |
| GroupKeyData | GROUP\_KEY\_DATA | (Optional) Encrypted group key. |
| VerifyGroupCode | VERIFY\_GROUP\_KEY | (Optional) Verification data for group key. |
| GroupStatus | GROUP\_STATUS | Status of the group operation. |

aSubgroupRange parameter shall be present for a fragmented GKB.

bThe UserSpecificData parameter can be used to convey additional information such as version information of the GKB used or additional credentials.

cIf CompleteSubree is Null, no Node Index is matched the CompleteSubtree.

dIf ComplementSubtreeFlag is not present, it means ComplementSubtreeFlag = 0.

* Revise 7.4.32.1 MIH\_Net\_Group\_Manipulate.request

MIH\_Net\_Group\_Manipulate.request (

DestinationIdentifier,

ResponseFlag,

GroupKeyUpdateFlag,

TargetIdentifier,

TransportAddress,

MasterGroupKey,

SubgroupRange,

UserSpecificData,

CompleteSubtree,

ComplementSubtreeFlag,

GroupKeyData,

VerifyGroupCode

)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| DestinationIdentifier | MIHF\_ID | Specifies group MIHF-ID of the remote MIHF peers. DestinationIdentifier may be different from TargetIdentifier. |
| ResponseFlaga | RESPONSE\_FLAG | (Optional) Flag that represents whether or not a response is needed. |
| GroupKeyUpdateFlag | GROUP\_KEY\_UPDATE\_FLAG | Flag that represents whether or not a group key in GroupKeyData is updated. |
| TargetIdentifier | MIHF\_ID | The target MIHF group identifier for the group operation. |
| MulticastAddress | TRANSPORT\_ADDR | (Optional) Multicast address corresponding with the target group identifier. |
| MasterGroupKey | MGK | (Optional) The master group key associated with the target MIHF group identifier. |
| SubgroupRange | SUBGROUP\_RANGE | (Optional) Subgroup to process the command a |
| UserSpecificData | OCTET\_STRING | (Optional) Auxiliary data. |
| CompleteSubtree | COMPLETE\_SUBTREE | (Optional) Complete Subtree data **c**. |
| ComplementSubtreeFlag | SUBTREE\_FLAG | (Optional) Flag to interpret the complete subtree data (See 9.5.2) |
| GroupKeyData | GROUP\_KEY\_DATA | (Optional) Encrypted group key. |
| VerifyGroupCode | VERIFY\_GROUP\_KEY | (Optional) Verification data for group key. |

a In case the ResponseFlag parameter is not present, the MIHF shall generate a request message, and otherwise the MIHF generates either a request or an indication message, based on the ResponseFlag parameter.
b SubgroupRange parameter shall be present for a fragmented GKB.
c In case the GroupKeyData parameter is not present, the CompleteSubtree parameter shall be present.

* Revise 8.6.1.24 MIH\_MN\_Group\_Manipulate response

|  |
| --- |
| **MIH Header Fields (SID=1, Opcode=2, AID=11 )** |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| TargetIdentifier(Group Identifier TLV) |
| SequenceNumber (conditional)ª(Sequence Number TLV) |
| TransportAddress (Optional)(Transport Address TLV) |
| SubgroupRange (Optional)(Subgroup\_Range TLV) |
| UserSpecificData (Optional)(Aux Data TLV) |
| CompleteSubtree (Optional)(Complete Subtree TLV) |
| ComplementSubtreeFlag (Optional)b(Complement Subtree Flag TLV) |
| GroupKeyData (Optional)(Group Key Data TLV) |
| GroupStatus(Group Status TLV) |
| VerifyGroupCode (Optional)(Verify Group Code TLV) |
| SecurityAssociationID (Optional) (SAID Notification TLV) |

ª This parameter is carried for CCM counter synchronization purpose when GroupKeyData is carried.

b If ComplementSubtreeFlag is not present, it means ComplementSubtreeFlag = 0.

* Revise 8.6.1.25 MIH\_Net\_Group\_Manipulate request

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=1, AID=12 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| TargetIdentifier(Group Identifier TLV) |
| SequenceNumber (Optional)a(Sequence Number TLV) |
| MulticastAddress (Optional)(Multicast Address TLV) |
| SubgroupRange (Optional)(Subgroup Range TLV) |
| UserSpecificData (Optional)(Aux Data TLV) |
| CompleteSubtree (Optional)(Complete Subtree TLV) |
| GroupKeyData (Optional)(Group Key Data TLV) |
| VerifyGroupCode (Optional)(Verify Group Code TLV) |
| SecurityAssociationID (Optional) (SAID TLV) |

a This parameter is only used in the case CCM encryption method is used and the group key is not updated.

* Revise 8.6.1.26 MIH\_Net\_Group\_Manipulate indication

|  |
| --- |
| MIH Header Fields (SID=1, Opcode=3, AID=12 ) |
| **Source Identifier** = sending MIHF ID(Source MIHF ID TLV) |
| **Destination Identifier** = receiving MIHF ID(Destination MIHF ID TLV) |
| TargetIdentifier(Group Identifier TLV) |
| SequenceNumber (Optional)(Sequence Number TLV) |
| MulticastAddress (Optional)(Multicast Address TLV) |
| SubgroupRange (Optional)(Subgroup Range TLV) |
| UserSpecificData (Optional)(Aux Data TLV) |
| CompleteSubtree (Optional)(Complete Subtree TLV) |
| GroupKeyData (Optional)(Group Key Data TLV) |
| VerifyGroupCode (Optional)(Verify Group Code TLV) |
| SecurityAssociationID (Optional) (SAID TLV) |

* Revise Table F.24- Data type for security

|  |  |  |
| --- | --- | --- |
| COMPLETE\_SUBTREE | CHOICE(LIST (GKB\_INDEX), COMPLETE\_SUBTREE\_BLOOM\_FILTER) | The data type for the complete subtree part of a GKB.  |
| COMPLETE\_SUBTREE\_BLOOM\_FILTER  | SEQUENCE(OCTET\_STRING, UNSIGNED\_INT(1)) | The OCTET\_STRING part contains a Bloom Filter value computed against a set of GKB\_INDEX in Complete Subtree. The UNSIGNED\_INT(1) part contains Bloom Filter parameter k. See Annex N for detailed operations. |
| VERIFY\_GROUP\_KEY | SEQUENCE ( OCTETS(16), OCTETS(16)) | The first OCTET(16) is arbitrary data, which is an input message to AES-CMAC (defined in RFC-4493). The second OCTET(16) is the MAC value for the first OCTET(16) to be verified. |

* Revise Table L.2- Type values for TLV encoding

|  |  |  |
| --- | --- | --- |
| TLV type name | TLV type value | Data Type |
| Verify Group Code | 100 | VERIFY\_GROUP\_KEY |

* Revise 9.6.1 Group session key derivation
	+ Change the procedure to generate the key for VerifyGroupCode.

**9.6.1 Group session key derivation**



**—Figure 43: Key derivation example**

When a recipient of a GKB successfully decrypts a MGK from the GKB, a media-independent group session key (MIGSK) is derived from the MGK to protect group manipulation commands and group addressed commands:

For the key derivation, the following notations and parameters are used.

*K*: key derivation key. It is truncated from a master group key (MGK). The length of *K* is determined by the pseudorandom function (PRF) used for key derivation. If HMAC-SHA-1 or HMAC-SHA-256 is used as a PRF, then the full MGK is used as key derivation key, *K*. If CMAC-AES is used as a PRF, then the first 128 bits of MGK are used as derivation key, *K*.

*L*: The binary length of derived keying material MIGSK and MIGKVK. *L* is determined by selected group ciphersuite described in 9.6.4.

*h*: The output binary length of PRF used in the key derivation. That is, *h* is the length of the block of the keying material derived by one PRF execution. Specifically, for HMAC-SHA-1, *h* = 160 bits; for HMAC-256, *h* = 256 bits; for CMAC-AES, *h* = 128 bits.

*n*: The number of iterations of PRF in order to generate *L*-bits keying material.

*c*: The group ciphersuite code is a one octet string specified for each ciphersuite. The code is defined in 9.6.4.

*v*: The length of the binary representation of the counter and the length of keying material L. The default value for *v* is 32.

“MIGSK”: 0x4D4947534B, ASCII code in hex for string “MIGSK.”

[a]2: Binary representation of integer *a* with a given length.

For given PRF, the key derivation for MIGSK and MIGKVK can be described in the following procedures:

**Fixed input values**: h and v.

**Input**: *K*, *L*, and group ciphersuite code.

**Process**:

1. 
2. If *n* > 2v-1, then indicate an error and stop.
3. Result(0) := empty string.
4. For i = 1 to *n*, do
	1. *K*(i) := PRF(*K*, “MIGSK” || [i]2 || *c* || [*L*]2).
	2. Result(i) = Result(i-1) || *K*(i).
5. Return Result(n) and MIGSK is the leftmost *L* bits of Result(n).

**Output**: MIGSK || MIGKVK.

With the above procedure, a key hierarchy is derived as shown in Figure 43.

This mechanism conforms with NIST SP800-108 (KDF in Counter Mode).

* Add following texts to 9.6.5 Group key distribution Ciphersuites
	+ Add explanation texts for generation GroupKeyVerificationCode.

9.6.5 Group key distribution Ciphersuites

The ciphersuites used for distributing a master group key are defined in Table 7.

1. –Group key distribution Ciphersuites

|  |  |  |
| --- | --- | --- |
| Code | Wrapping Algorithm | MAC AlgorithmFor VerifyGroupCode |
| 11010100 | AES\_Key\_Wrapping-128 | NULL |
| 11000100 | AES\_ECB-128 | NULL |
| 11000101 | AES\_ECB-128 | AES-CMAC-128 |
| 11000000 | No group key distribution | NULL |

In Table 7, AES\_Key\_Wrapping is an AES mode of operations specified in NIST SP 800-38F.

Note that ECB mode is not recommended to protect a key, because it cannot provide proper security level for the key. In particular, the same plaintext will be encrypted to the same ciphertext, since no random IV is used for each encryption. On the other hand, if transmitting IVs will increase the size of GroupKeyData to an unacceptable point for the transport protocol, then ECB mode may be used, assuming that the same key will be retransmitted with a very small probability and signature is applied to provide authentication and integrity.

The support of code “11010100” is mandatory and all entities supporting this specification shall implement it.

Note that digital signature algorithm ECDSA-256 is used to protect group key distribution.

* Add following texts to 9.5.2.1 Master Group Key Wrapping
	+ Add following texts in line 40, p 59: how to generate VerifyGroupCode.

The MIHF selects a string, and derives MIGKSK from MGK as described in 9.6.1.

The MIHF derives MAC of the string using MIGKSK. The sting and MAC become VerifyGroupCode for MGK. Figure XX presents a flow diagram of the verify group code generating procedure.



**Figure XX: Flow diagram of the verify group code generation**

* + Revise the texts in line 1, p 60:

If a master group key is not distributed in the GKB, the master group key wrapping step and the verify group code generating step should be omitted. If an authenticated encryption (e.g., AES-Key\_Wrapping) is used to generate GroupKeyData, the verify group code generating step can be omitted.

After generating the GroupKeyData, The MIHF can replace the list of Node Indices in the Complete Subtree by the Bloom Filter of the list. If the MIHF uses the Bloom Filter as the Complete Subtree, the GroupKeyData shall be generated by the authenticated encryption or the VerifyGroupCode shall be attached.

* Revise 9.5.2.2 and Figure 36
	+ Add explanation texts 1) in case CompleteSubtree is not attached, 2) in case CompleteSubtree is represented by Bloom Filter.

 **9.5.2.2 Master group key unwrapping**

 The master group key unwrapping procedure for a GKB ~~with CompleteSubtree and GroupKeyData~~ is described as follows:

 An MIHF in the recipient of GKB checks GroupKeyData. If GroupKeyData is contained, go to mastergroup key unwrapping procedure; else start no group key procedure.



Figure Y: Flow diagram of …

If Group Key Data is contained, the MIHF in the recipient of GKB checks CompleteSubtree. If CompleteSubtree contains a list of Node Index, go to master group key unwrapping procedure 1 in 9.5.2.2.1. Else if CompleteSubtree contains a Bloom Filter, go to master group key unwrapping procedure 2 in 9.5.2.2.2. Else if CompleteSubtree is not given, go to master group key unwrapping procedure 3 in 9.5.2.2.3.



Figure X: Flow diagram of the group key unwrapping

**9.5.2.2.1 No group key data procedure**

In case a CompleteSubtree is given as a list of node index and a GroupKeyData is not given, no group key procedure can be executed.

The no group key procedure is described as follows:

1. The procedure reads a CompleteSubtree S.
2. The procedure reads a list of Node Indices *I* for device keys of the recipient.
3. The procedure finds *n* in *I* such that *n* is *i*-th element of the complete subtree S. If it succeeds to find such *n*, it returns *Success*. Else it returns *Fail*.

Figure Z is a flow diagram of the master group key unwrapping procedure 1.



Figure Z: Flow diagram of no group key data procedure

**9.5.2.2.2 Master group key unwrapping procedure 1**

In case a CompleteSubtree is given as a list of node index and a GroupKeyData is given, Master group key unwrapping procedure 1 can be executed.

The master group key unwrapping procedure 1 is described as follows:

1. The procedure reads a CompleteSubtree S and a GroupKeyData KB.
2. The procedure reads a list of Node Indices *I* for device keys of the recipient.
3. The procedure finds *n* in *I* such that *n* is *i*-th element of the complete subtree S. If it succeeds to find such *n*, then it extracts a device key k that corresponds to n. Else it returns *Fail*.
4. The procedure reads an encrypted group key C from *i*-th element of KB.
5. The procedure decrypts C using k. If the decryption is successful, it returns *Success* and the result of the decryption as a master group key, MGK. Else it returns *Fail*.

Figure X1 is a flow diagram of the master group key unwrapping procedure 1.



Figure X1: Flow diagram of the master group key unwrapping procedure 1

**9.5.2.2.3 Master group key unwrapping procedure 2**

In case a CompleteSubtree is given as a Bloom Filter and a GroupKeyData is given, Master group key unwrapping procedure 2 can be executed.

The master group key unwrapping procedure 2 is described as follows:

1. The procedure reads a Bloom Filter *BF* in a CompleteSubtree, a GroupKeyData *KB*, andVerifyGroupCode *VGC* (an option).
2. The procedure reads a list of node indices *I* for device keys of the recipient, and sets *i* = 1.
3. The procedure sets *n* to to *i*-th element of *I*.
4. The procedure checks whether *n* matches *BF*. If it succeeds to the matching, then it extracts a device key *k* that corresponds to *n,* and it sets *j* = 1. If it fails to the matching, go to Step g).
5. The procedure sets *C* to *j*-th element of *KB*.
6. The procedure decrypts *C* using *k*. The MIHF checks the result of decryption using VerifyGroupCode (an option). If the decryption is successful, it returns *Success* and the result of the decryption as a master group key, *MGK*. Else if *C* is not the last element of *KB*, it sets *j* = *j*+1, and goes to Step e).
7. If *n* is not the last element of *I*, the procedure sets *i* = *i*+1 and goes to Step c). Else it returns *Fail*.



Figure X2 is a flow diagram of the master group key unwrapping procedure 2.

**9.5.2.2.4 Master group key unwrapping procedure 3**

In case a CompleteSubtree is not given and a GroupKeyData is given, Master group key unwrapping procedure 3 can be executed.

The master group key unwrapping procedure 3 is described as follows:

1. The procedure reads a GroupKeyData *KB*, andVerifyGroupCode *VGC* (an option).
2. The procedure reads a list of device keys *DK* of the recipient, and sets *i* = 1and *j* =1.
3. The procedure sets *k* to to *i*-th element of *DK*.
4. The procedure sets *C* to *j*-th element of *KB*.
5. The procedure decrypts *C* using *k*. The MIHF checks the result of decryption using VerifyGroupCode (an option). If the decryption is successful, it returns *Success* and the result of the decryption as a master group key, *MGK*. Else if *C* is not the last element of *KB*, it sets *j* = *j*+1, and goes to Step d).
6. If n is not the last element of *I*, the procedure sets *i* = *i*+1 and goes to Step c). Else it returns *Fail*.



Figure X3 is a flow diagram of the master group key unwrapping procedure 3.

* Revise 9.5.3.1.1 Group Manager and Figure 38
1. (Optional) Generate GroupKeyData and VerifyGroupCode, and set CompleteSubtree:
	1. When MGK is not distributed, this process is skipped.
	2. Send the MGK and the CompleteSubtree to the MasterGroupKeyWrapping procedure, and receive GroupKeyData. The procedure accesses the *Tree Information Base* to refer all the pairs of a Node Index and a corresponding node key.
	3. Set MGK to MasterGroupKey.
	4. (Optional) Send the MGK to the verify group code generating procedure, and receive VerifyGroupCode.
	5. (Optional) Set CompleteSubree to Bloom Filter of a list of NodeIndices in the CompleteSubtree.
	6. (Optional) Delete CompleteSubtree.



Generate VerifyGroupCode

VerifyGroupCode?

Delete CompleteSubtree

Delete?

Set CompleteSubtree as BloomF ilter

Bloom Filter?

# **Figure 38— Summary of steps performed by MIH User of PoS with group manager**

Revise 9.5.3.1.2 MIHF of a PoS with Group Manager

 Insert following step after Step e).

1. If the MIH\_Net\_Group\_Manipulate.request contains a VerifyGroupCode, it generates a Verify Group Code TLV from the VerifyGroupCode.



Add “Group Identifier TLV,”

“(Optional) VerifyGroupCode TLV”

Add “Sequence Number TLV”

Remove

Please move to here.

Remove
(These steps are shown in 9.6.2.)

* Revise 9.5.3.1.3 Receiving procedures for group manipulation commands
	+ Revice Step b)
1. The MIHF processes the Complete Subtree in the Complete Subtree TLV, ~~and~~ a GroupKeyData in the Group Key Data TLV, and a VerifyGroupCode in the Verify Group Code TLV as described in 9.5.2.2. If the MIHF succeeds to find a matching pair of Node Indices, go to Step c). Otherwise, go to Step d).

****

Process Complete Subtree TLV, Group Key Data TLV, and Verify Group Code TLV by the procedure in 9.5.2.2