**IEEE P802.21
Media Independent Handover Services**

|  |
| --- |
| **Proposed Remedy for the 802.21d LB7 comment #196** |
| **Date:** 2013-09-16 |
| **Author(s):** |
| **Name** | **Affiliation** | **Address** | **Phone** | **Email** |
| Yoshikazu Hanatani | Toshiba |  |  | yoshikazu.hanatani@ toshiba.co.jp |

Abstract

This document contains proposed remedy for “the 802.21d ballot 7 comment #196 about a relation between a group key in Annex P and a master group key in Section 9.4.5.

# Remedy for the 802.21d LB7 comment #196

# Annex U(informative)GKB toy example

An example is introduced to explain the basic principle of GKB and how to make a GKB. Consider a binary tree of Depth 4. The nodes other than the root node are labeled ‘0’, ‘1’, ‘00’, ‘01’, ‘10’, ‘11’, …, ‘0000’, ‘1111’, up to down and left to right. (See Figure U.1).

The label is sometimes called Node Index. A Node Index assigned to a leaf is especially called Leaf Number. Each node is assigned a key: k(0), k(00), k(01), …, k(0000), k(0001), …, k(1110), k(1111). Let the keys be called Node Keys. An MN is associated with a unique leaf. Thus, sixteen MNs are associated with the leaves of the tree: Call them ‘MN0’, ‘MN1’, …, ‘MN15’, left to right. Each MN is assigned a set of pairs of a Node Index and a Node Key, which is called Device Key: An MN is assigned the pairs along the path that is descending from the root to the leaf associated with the MN. For instance, MN3 is assigned the following Device Key: {(0, k(0)), (00, k(00)), (001, k(001)), (0011, k(0011))}.

Figure .— An Example Tree

A set of MNs is called group if and only if they share an MIHF Group ID and a group key. A group key in Annex U (i.e., Ka,1, Ka,2, Kb) is a master group key (MGK) in section 9.4.5.

At first, make all the sixteen MNs constitute one group, say, Group AG. Then, make the GKB such that {{0, 1}, {<k(0)>[Ka,1], <k(1)>[Ka,1]}}, where Ka,1 is the group key for Group A and <k>[D] denotes data D encrypted by a key k. {0, 1} is the complete subtree part of the GKB and {<k(0)>[Ka,1], <k(1)>[Ka,1]} is the group key data part. Check if all the MNs can share the group key. Any Device Key has one of the Node Keys: k(0) or k(1). Therefore, any MN can decrypt the preceding GKB to derive the group key Ka,1. The group key is shared by all the MNs as expected.

Let MN1, MN4 and MN5 be removed from Group A: Then the GKB will cover {MN0, MN2, MN3, MN6, …, MN15}. The GKB required for this is as follows:

GKB1 = {{1, 001, 011, 0000}, {<k(1)>[Ka,2], <k(001)>[Kaa,2], <k(011)>[Ka,2], <k(0000)>[Ka,2]}},

where Ka,2 is a new group key for Group A. Check that any MN in Group A can decrypt one of the elements of the group key data part and derive the group key. Also note that the complete subtree part of the GKB is ordered in the ascending dictionary order defined in 9.4.2.1. And, let Group B be a group which is composed of MN3, MN4, MN8, MN9 and MN12, MN13, MN14 and MN15. The GKB to create Group B is the following GKB2:

GKB2 = {{11, 100, 0011, 0100}, {<k(11)>[Kb], <k(100)>[Kb], <k(0011)>[Kb], <k(0100)>[Kb]}},

where Kb is a group key for Group B. Note that multiple groups with their own group keys may exist on one tree. An MN with one Device Key Set may belong to multiple groups at the same time.

It may be that the size of a GKB is too large for an MIH service specific TLV. The followings show one example of the ways how a GKB is fragmented into smaller pieces: Suppose here that the capacity of the TLV allows only two encrypted group keys. GKB2 can be fragmented into two GKBs: GKB2-1 and GKB2-2 such that GKB2-1 = {{11, 100}, {<k(11)>[Kb], <k(100)>[Kb]}} and GKB2-2 = {{0011, 0100}, {<k(0011)>[Kb], <k(0100)>[Kb]}}. Suppose that GKB2-2 arrives at MN15 after GKB2-1 does first. MN15 joins in Group B when it receives GKB2-1. Then, MN15 leaves Group B receiving GKB2-2 because it has no key to successfully decapsulate GKB2-2. This is clearly not the expected behavior for MN15. This problem can be avoided if the Subgroup Ranges of the GKBs are appropriately set. Let GKB2-1 = {R1, {11, 100}, {<k(11)>[Kb], <k(100)>[Kb]}} and GKB2-2 = {R2, {0011, 0100}, {<k(0011)>[Kb], <k(0100)>[Kb]}}, where R1 = [8, 15] and R2 = [0, 7]. The Leaf Number of MN15 is 15, which means that it is in the range of R1. Thus, it processes GKB2-1 and derives the group key. MN15, however, does not process GKB2-2 because it is out of the range of R2. Thus, MN15 joins in Group B and stays there as expected.

There is a version of GKB without a group key data part, which are used when confidentiality is not necessary for group commands. Creation of such a GKB is the same. For instance, just remove the GroupKeyData field from a GKB having a group key data part.