A Secure Multicast Group Management and Key Distribution in IEEE 802.21

Yoshikazu Hanatani[1], Naoki Ogura[1], Yoshihiro Ohba[2], Lidong Chen[3], and Subir Das[4]

[1] Toshiba Corp., 1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki, 212-8582, Japan.
{yoshikazu.hanatani, naoki.ogura}@toshiba.co.jp
[2] Toshiba Electronics Asia Pte.,Ltd., 20 Pasir Panjang Road, #12-25/28 Mapletree Business City, Singapore, 117439.
yoshihiro.ohba@toshiba.co.jp
[3] National Institute of Standards and Technology, 100 Bureau Dr. Gaithersburg, MD 20899-8930, U.S.A.
lily.chen@nist.gov
[4] Applied Communication Sciences, 150 Mount Airy Road, Basking Ridge, New Jersey, U.S.A., 07920.
sdas@appcomsci.com

Abstract. Controlling a large number of devices such as sensors and smart end points, is always a challenge where scalability and security are indispensable. This is even more important when it comes to periodic configuration updates to a large number of such devices belonging to one or more groups. One solution could be to take a group of devices as a unit of control and then manage them through a group communication mechanism. An obvious challenge to this approach is how to create such groups dynamically and manage them securely. Moreover, there needs to have mechanisms in place by which members of the group can be removed and added dynamically. In this paper, we propose a technique that has been recently standardized in IEEE 802.21 (IEEE 802.21-2015™) with the objective in providing a standard-based solution to the above challenges. The approach relies on Logical Key Hierarchy (LKH) based key distribution mechanism but optimizes the number of encryption and decryption by using "Complete Subtree". It leverages IEEE 802.21 framework, services, and protocol for communication and management. It provides a scalable and secure way to manage (e.g., add and remove) devices from one or more groups. We describe the group key distribution protocol in details and provide a security analysis of the scheme along with some performance results from a prototype implementation.

Keywords: Group Communication, group key and management, multicast, Group Key Block (GKB), Subtree, IEEE 802.21

1 Introduction

In today's networked world, it is becoming more and more expensive when it comes to configuring and software updates to large number of remote sensors and smart devices. To alleviate the cost and scalability issues, operators and vendors perform these operations remotely, commonly known as remote device management. While remote configurations updates are very common and secure networking technologies are available, normally it happens via a remote server in which each device requires to connect the server. This process becomes bandwidth inefficient (*n* unicast connections) and time consuming when the configuration update of a group of devices involves transferring a large amount of data. On the contrary, if these updates can be performed via a secure group communication mechanism whereby the network entity can multicast or broadcast the messages to a group of devices, the process becomes more efficient and saves a great deal of time and network resources,

IEEE 802.21-2008™ [1] defines a media independent framework, services and signaling protocol that are standardized in IEEE while the transport of the signaling protocol over IP is standardized in IETF. The standard published in [1] addresses the handover optimization use case whereby the user experience of ongoing application flows can be improved significantly for mobile nodes (MNs) that are moving from one link layer access technology to another irrespective of whether the access network is managed by the same or different network operators. The framework provides a signaling protocol that can be transported natively over the link layer or over Internet Protocol (IP) using underlying unicast and multicast mechanisms. In subsequent years of Standards amendment process, IEEE 802.21 Working Group addressed other use cases and defined signaling protocol and services security along with a group management mechanism in [2, 3]. In particular, Standards published in [3] targeted the use case where a large number of groups of devices are required to be managed from a group manager that resides in an entity in the network. Therefore [3] is relevant to our discussion in which a network entity can multicast a message to a group of nodes (or devices) using IEEE 802.21 media independent protocol interface, and secure group key distribution mechanism to cryptographically protect these multicast messages. The amendment [3] not only adds the secure group communication mechanism but also allows network nodes to communicate handover messages and to perform other management operations such as failover, failback, and configuration updates to a group of devices that are part of the network. The standardized approach relies on Logical Key Hierarchy (LKH) based key distribution mechanism [4, 5, 6] and uses "Complete Subtree" to optimize the number of encryption.

In this paper, we first introduce IEEE 802.21 [3] defined protocol and then discuss how to use the complete subtree method to optimize the performance of group communication. Subsequently, we introduce specific methods to handle the issues in group key distribution for IEEE 802.21 applications. In addition, we also analyze security of the group key distribution protocol as specified in IEEE 802.21 [3].

The paper is organized as follows: Section 2 discusses the related work. Section 3 presents the preliminaries of group key distribution approach while Section 4 describes the Group Key Block. Section 5 describes the group key distribution scheme and Section 6 provides a formal model-based security analysis. Section 7 captures our initial prototype implementation results and Section 8 concludes the paper.

2 Related work and our approach

Secure multicast-based communication has been an important research topic in cryptography and in communication security. Most of the research discusses theoretical boundaries on the message length (i.e., number of encryptions), storage (i.e., number of keys each member holds), and computations for each receiver [7]. Some of the research also discusses trace-and-revoke algorithm with an upper bound of coalitions, which is outside the scope of this paper.

In practical applications, the secure group communications have been handled through initial pairwise group key distribution to group members [8, 9]. The schemes in [8] allow group key distribution for rekeying. On the other hand, whenever new members join the group or some current members leave the group, the schemes defined in [9] have to use pairwise secure channels for key distribution.

Logical Key Hierarchy (LKH) has been introduced in [4, 5, 6] for group key update, assuming each group member has been provisioned with one fixed individual key or the individual keys are established using other methods. The LKH is represented as a tree while the individual keys are represented as leafs of the tree. The nodes above the leaf level represent the keys shared by different members represented as leafs which have a path to the node. Every time, a member or members join or leave the group, the tree is updated.

The group key distribution scheme introduced in this paper uses a similar tree to represent the fixed keys that each group member hold. The group key is encrypted by a set of keys represented in the tree such that each member in the group owns a key to decrypt it, while the nodes not in the group do not have the proper decryption keys. Each time when group members join or leave, a new group key is distributed using the proper keys for the new group. Intuitively, in a given group, if more group members shared the same key, that is, their paths meet at the same node, the less encryptions are needed. In order to gain such efficiency, the scheme in this paper uses 'Complete Subtree'. The 'Complete Subtree' method is introduced in [10] to optimize the number of encryptions and decryptions for each group key distribution. These methods have not been adopted in the practical applications to the best of our knowledge.

In particular, the method in this paper uses a single key tree to distribute keys for different groups. For a given group, our method generally requires less number of key encryptions than LKH for the key distributor, which also means lower transmission burden. Let L (> 1) denote the number of leaf nodes of the key tree, $N(< L)$ denote the number of root nodes of complete subtrees covering all leaf nodes of the members of the group, and M denote the number of ancestor nodes of the N root nodes. In initial group key distribution, LKH requires at least L encryptions to distribute the group key (which is the key corresponding to the root node of the key tree in LKH) and other keys to be used for key update. In group key update, LKH requires $(N + M)$ encryptions to update the group key for all group members excluding revoked members. In contrast, our method requires N encryptions of the group key for both initial group key distribution and group key update. Therefore, if we assume the same key tree size, our method always requires less number of encryptions than LKH. Our method allows to take advantage of complete subtrees when possible. That is, when the group members represented by the leaf nodes can be grouped to complete subtrees, the number of key encryptions can be further reduced.

For any group member m, our method requires a single decryption to obtain the group key, while for LKH, $(H - H_m + 1)$ decryptions are needed where H represents the height of the key tree and H_m represents the height of the complete subtree that covers the leaf node of member m. In the applications where each group member is a constrained device such as a sensor, our method has significant advantages.

On the other hand, it has not been clear how scalable complete subtree is and no specific algorithms have been proposed to identify the complete subtrees. The use of a media independent framework and a signaling protocol that can be transported natively over Ethernet or over Internet Protocol (IP) using underlying unicast and multicast mechanisms is another important aspect that has not been standardized or published to the best of our knowledge.

While different security notions for group key agreement protocols have been introduced in [11, 12, 13, 14], we define a variant of another formal security model called Bresson and Manulis (BM) model [15] to satisfy the similar security requirements. The BM model cannot be applied directly for our security proof because the group key distribution protocol specified in [3] does not provide perfect forward secrecy.

3 Preliminaries

In this section, we introduce some basic concepts used in group management and key distribution. The concepts of key tree and complete subtree are essential for the key distribution protocol that we discuss in subsection 3.1. When a group key is distributed, it is protected by a key wrapping mechanism. To authenticate the sender, the encrypted group key is digitally signed. The security notions and definitions of key wrapping and signatures are introduced in subsections 3.2 and 3.3, respectively.

3.1 Key tree and complete subtree method

In this paper, we assume there are a large set of devices $\mathcal{U} = \{U_1, U_2, \ldots, U_m\}$ in which each device is provisioned with a set of keys, called *device keys*, DK_i. In the group key distribution protocol, the key is distributed by a group manager (GM) to a subset of the devices $S = \{U_{i1}, U_{i2}, \ldots, U_{ik}\}$.

A key tree is a binary tree with depth n and it has t levels from the root to the leafs. Therefore, such a key tree has 2^n leaf nodes whereby each leaf node is a device and to represent all the devices $\{U_1, U_2, \ldots, U_m\}$, it requires $m \leq 2^n$. **Figure 1**~~**Figure 1**~~ is an example of depth-3 tree. Each node (e.g., a leaf node, an inner node, or the root node) is coded with a binary string called *index* and a key. Assume the root node is on the top. The next level nodes have indices 0 and 1 from the left to the right. The corresponding keys are denoted k_0 and k_1. The next level nodes have indices 00, 01 as decedents of node 0 while 10 and 11 as decedents of node 1. The corresponding keys are denoted k_{00}, k_{01}, k_{10}, k_{11}. Nodes in every level are indexed this way until the level t. In the rest of this paper, we will denote the node with the index and the key (I_i, k_i). We simply call each key as a node key labeled with its index. For the leaf node, we also use the integer converted from its index as the leaf number which maps to a specific device.

For device U_j, the provisioned device keys consist of all the node keys from the leaf along the path to the root. In the example depicted in **Figure 1**~~**Figure 1**~~, a device represented by leaf "000" is provisioned with device keys {k(root), k_0, k_{00}, and k_{000}}.

In order to distribute a master group key to the devices in a specific group, the master group key mgk is protected with a set of keys in such a way that each device in the group must own a key in its device key set to recover the mgk, while for any device not in the group, it cannot recover the mgk. For a given group, there must be many different ways to protect the mgk. For example, in **Figure 1**~~**Figure 1**~~ consider a group represented by the leaf nodes 000, 001, 010, 011, 101, and 111. Notice that nodes 000 and 001 share the same key k_{00} and nodes 000, 001, 010, 011 all share the key k_0, then protection with the following key sets all satisfy the condition stated above.

a. k_{000}, k_{001}, k_{010}, k_{011}, k_{101}, k_{111};
b. k_{00}, k_{01}, k_{101}, k_{111};
c. k_0, k_{101}, k_{111}.

Obviously, key set c is more appealing because it calls the least number of the protection mechanisms and thus, generated the shortest of the ciphertext for broadcast. The concept of complete subtree is introduced to optimize the number of calls to the protection mechanisms. In this paper, the protection mechanisms can be a key wrapping algorithm or an encryption algorithm.

A complete (depth-l) subtree in a depth-t tree is a subtree with 2^l leaf nodes such that their indices have common prefix of t-l bits. For the tree in **Figure 1**~~**Figure 1**~~ , nodes represented with indices 000 and 001 form a depth-1 complete subtree at root 00, while nodes represented with indices 000, 001, 010, and 011 form a depth-2 complete subtree at root 0. For a subset of the group, if it can form a complete subtree, using the key represented by the subtree root allows all the members to recover the protected group key. Therefore, identifying complete subtrees in a given group can optimize the computation and communication resources in group key distribution for that group.

It shall be noticed that a single leaf is a depth-0 complete subtree. In fact, the optimization is to find out the non-overlapping maximum complete subtrees, which can cover the whole group. The set of non-overlapping maximum complete subtrees is unique for a given group. For example, for the group with the leaf nodes 000, 001, 010, 011, 101, 111, the set of the non-overlapping maximum complete subtrees that covers all the members is a depth-2 complete subtree and two depth-0 complete subtrees. Standards published in [3] specifies a complete subtree algorithm to determine such set.

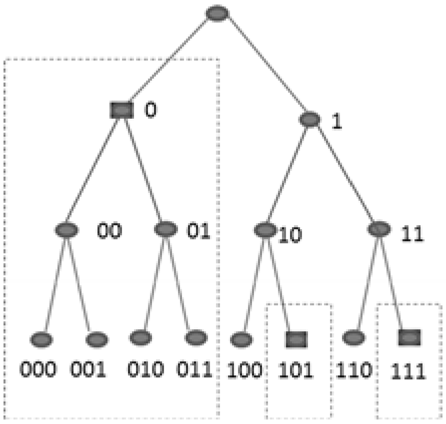

Figure 1. A depth-3 key tree

3.2 Key-wrapping scheme

After determining which keys to use through the complete subtrees, for the group key distribution, the key is protected by a key wrapping scheme. Key-wrapping scheme is a symmetric key encryption scheme for sending a group key. For group key distribution, IEEE std 802.21d-2015™ [3] supports two deterministic symmetric key schemes, AES-key-wrapping-128 and AES-ECB-128. Here $x \longleftarrow_R X$ means that x is an element chosen uniformly at random in a finite set X.

Definition 1. *Key-wrapping* $\mathcal{KW}$ *is a 3-tuple of algorithms* $(\mathsf{KeyGen}_{\mathcal{KW}}, \mathsf{Wrap}, \mathsf{Unwrap})$ *satisfying*:

- $\mathsf{KeyGen}_{\mathcal{KW}}$*: a probabilistic algorithm takes the security parameter* κ*, and returns* $K \in \{0,1\}^{\kappa}$*,*
- Wrap *: a deterministic algorithm takes* $K \in \{0,1\}^{\kappa}$*and* $D \in \{0,1\}^{l}$*, and returns* $C \in \mathcal{C}$ *where* l *is a bit-length of key to be wraped.*
- Unwrap *: a deterministic algorithm takes* $K \in \{0,1\}^{\kappa}$ *and* $C \in \mathcal{C}$*, and returns* $D \in \{0,1\}^{l} \cup \{\perp\}$*,*
  where $\forall K \leftarrow \mathsf{KeyGen}_{\mathcal{KW}}(\kappa), \forall D \in \{0,1\}^{l}$: $\mathsf{Unwrap}(K, \mathsf{Wrap}(K,D)) = D.$

A basic security requirement for symmetric encryption scheme is the indistinguishability against chosen plaintext attack (IND-CPA), and it is well-known that no deterministic encryption schemes can satisfy the IND-CPA. On the other hand, for sending a random key, the following weaker security requirement is sufficient.

Definition 2. *(Indistingutishability against Random-Plaintext Attack) Let* $b \leftarrow_R \{0,1\}$ *and* $W \leftarrow \mathsf{KeyGen}_{\mathcal{KW}}(\kappa)$ *where* κ *is a security parameter. The RPA-*

advantage of $\mathcal{A}$ can send queries to oracles Wrap_W *and* LR_W*.When the oracle* Wrap_W *receives a query,* Wrap_W *selects* $D \leftarrow_R \{0,1\}^l$*, and returns* $(D, \mathsf{Wrap}(W, D))$*. When the oracle* LR_W *receives a query,* LR_W *selects* $D_0, D_1 \leftarrow_R \{0,1\}^l$ *, and returns* $(D_0, D_1, \mathsf{Wrap}(W, D_b))$*. The RPA-advantage of $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathrm{kw.rpa}}_{\mathcal{A},\,\mathcal{KW}}(\kappa) = \Pr\left[\mathcal{A}^{\mathsf{Wrap}_W,\mathsf{LR}_W} \to 1 | b = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Wrap}_W,\mathsf{LR}_W} \to 1 | b = 0\right]$$

where $\mathcal{A}$ is a probabilistic polynomial-time algorithm that sends at most q queries to Wrap_W *and at most 1 query to* LR_W*.*

$\mathcal{KW}$ is IND-RPA secure if for all $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{kw.rpa}}_{\mathcal{A},\,\mathcal{KW}}(\kappa)$ *is negligible.*

Random-Plaintext attack is originally defined in [16]. In the original definition in [16], the adversary is not allowed to query the Wrap_W. According to a similar discussion in [16], we can show that an ECB (electronic code book) mode based on a random permutation with block length *n* is IND-RPA secure per Definition 2.

3.3 Signature scheme

In order to authenticate a sender, IEEE std 802.21d-2015™ [3] supports one digital signature scheme, ECDSA (Elliptic Curve Digital Signature Algorithm). Here $x \longleftarrow_R X$ means that x is an element chosen uniformly at random in a finite set X.

Definition 3. *Signature Σ is a 3-tuple of algorithms* $\left(\mathsf{KeyGen}_{\mathcal{KW}}, \mathsf{Sign}, \mathsf{Verif}\right)$ *satisfying:*

- KeyGen_Σ *: a probabilistic algorithm that takes the security parameter κ, and returns a pair of public key and secret key (pk, sk),*
- Sign *: a probabilistic algorithm takes sk and a message $m \in \{0,1\}^*$, and returns σ,*
- Verif *: a deterministic algorithm takes pk, m, and σ, and returns 0 or 1,*

where $\forall(pk, sk) \leftarrow \mathsf{KeyGen}_\Sigma(\kappa)$, $\forall m \in \{0,1\}^*$: $\mathsf{Verif}(pk, m, \mathrm{Sign}(sk, m)) = 1$.

Definition 4. *(Existential Unforgeability against Chosen Message Attacks) Let* $\Sigma = (\mathsf{KeyGen}_\Sigma, \mathsf{Sign}, \mathsf{Verif})$ *be a digital signature scheme, and* $(pk, sk) \leftarrow \mathrm{KeyGen}_\Sigma(\kappa)$*. When a signing oracle* Sign_{sk} *receives a query* $m \in \{0,1\}^*$ *, it returns* $\sigma = \mathrm{Sign}(sk, m)$*. The advantage of $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathrm{euf-cma}}_{\mathcal{A},\Sigma}(\kappa) = \Pr[\mathcal{A}^{\mathsf{Sign}_{sk}}(pk) \to (m^*, \sigma^*){:}\mathsf{Verif}(pk, m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{M}]$$

where $\mathcal{M}$ is the set of message queried to Sign_{sk} *and $\mathcal{A}$ is a probabilistic polynomial-time algorithm who sends at most q_S queries to* Sign_{sk}*.*

Σ is EUF-CMA secure if for all $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{euf-cma}}_{\mathcal{A},\,\Sigma}(\kappa)$ *is negligible.*

4 Group Key Block

Group Key Block (GKB) is a data format defined in IEEE std 802.21d-2015™ [3] for encoding a group key and other data associated with the group key. The following attributes are contained in a GKB:

- GroupKeyData: a list of octet strings, each of them contains the group key encrypted by using a distinct node key specified in 'CompleteSubtree'. Either AES_Key_Wrapping-128 or AES_ECB-128 is used for encrypting the group key.
- GroupIdentifier: an identifier of a group.
- CompleteSubtree: a list of node indices corresponding to root nodes of specific subtrees of the key management tree. See Section 4.1 for more details.
- SubgroupRange: a range of valid leaf identifiers in the 'CompleteSubtree'. A 'SubgroupRange' is used when a GKB is fragmented into multiple smaller pieces (see Section4.2).
- VerifyGroupCode: a pre-known octet string encrypted by the group key. A 'VerifyGroupCode' is used for checking whether the decrypted group key is the same as the one generated by the GM. 'VerifyGroupCode' may be used when AES_ECB-128 is used for group key encryption (Note that AES_Key_Wrapping has a built-in key verification mechanism).

Digital signature is added to each message carrying a GKB using the signature scheme described in Section 3.3.

4.1 Encoding complete subtrees

IEEE std 802.21d-2015™ [3] defines three methods for encoding 'CompleteSubtree'. In this section, the default encoding method is explained. In the default encoding method, a list of the node indices is contained in the 'CompleteSubtree' where each node index in the key management tree represents the root node of a distinct subtree covering only leaf nodes corresponding to members of the group. A node index consists of a depth in the key management tree and a subindex that is unique within the depth. In 'GroupKeyData', i-th string contains the group key encrypted by the node key corresponding to the i-th node index in the 'CompleteSubtree'.

4.2 GKB fragmentation

As described in Section 4.1, the size of CompleteSubtree in Method 1 is proportional to the number of subtrees encoded. Also, when a GKB is multicast and the number of recipient is large (e.g., thousands or more), it is difficult to reliably deliver the GKB to all recipients.

IEEE std 802.21d-2015™ [3] addresses this issue by defining a special fragmentation mechanism for fragmenting GKB. Unlike other general-purpose fragmentation mechanisms (e.g., IP fragmentation), a recipient of GKB does not have to receive all the fragments of a single complete GKB and reassemble into the original GKB data. The recipient can instead determine whether it is a member of the group and can obtain the group key by receiving one GKB fragment that contains in 'SubgroupRange' attribute.. Suppose a single complete GKB is fragmented into five GKB fragments with 'SubgroupRange' of each fragment set to (0,99), (100,199), (200,299), (300,399), and (400,499). A recipient whose leaf identifier is 250, when receiving the GKB fragment with 'SubgroupRange (200,299)', can determine whether it is a member of the group. Thus it can obtain the group key if it is a group member and can simply ignore other four GKB fragments.

5 Group key distribution protocol

IEEE std 802.21d-2015™ [3] defines an architecture and a group key distribution protocol that a group manager (GM) can use to communicate to group members via a multicast transport. The group key distribution protocol uses the 'Complete Subtree' method with a deterministic symmetric key encryption scheme and a digital signature scheme. In this section, we introduce a simplified version of the group key distribution protocol using an option that is described in [3]. In this section we refer a group member to a user.

Provisioning

IEEE std 802.21d-2015™ [3] assumes that a group manager and each user has device keys, which are also called long-term keys. The secure provisioning method is not defined in the standard.

1. Let 2^n be the number of (potential) users managed by the group manager GM, and let $\mathcal{U}$ be a set of all users. GM generates a key tree T with depth n, and assigns (I_i, k_i) to each node in T where I_i is a node index represented as a binary string of length between 1 to n and $k_i \leftarrow \mathsf{KeyGen}_{\mathcal{KW}}(\kappa)$ is a node key where i corresponding to the node index I_i . For digital signature, GM generates $(pk, sk) \leftarrow \mathsf{KeyGen}_{\Sigma}(\kappa)$.
2. For all user U_i in $\mathcal{U}$, GM assigns each user U_i to a leaf node in T. Let $Path_{U_i}$ be a set of node indices of nodes from the leaf node which is assigned to U_i along the path to the root node. GM assigns $DK_i = \{(I_j, k_j)\}_{I_j \in Path_{U_i}}$, to U_i as the long-term keys.
3. GM securely sends *pk* and DK_i to each of U_i.

Procedure of GM

1. Decide a set of group members S which is a target for group key distribution and a group identifier GI which identifies a group using the distributed group key.
2. Pick a current sequence number *SN* for GI.

3. Decide a destination group DG for the group key distribution message. GM is required to send the group key distribution message to all of its members S. For simplicity, we assume that DG includes S. A broadcast group BG including all users may be used as DG.
4. Select a master group key $\mathsf{mgk} \in \{0,1\}^l$ uniformly at random and select a security association identifier SAID which is an identifier of a group session key $\mathrm{gsk} = \mathrm{KDF}(\mathsf{mgk})$ where KDF is a key derivation function which is publicly shared.
5. Compute a list of indices CS from $\mathcal{U} \backslash S$ and T by 'Complete Subtree' method.
6. For all $I_i \in CS$, compute $c_i = \mathsf{Wrap}(k_i, \mathsf{mgk})$ where (I_i, k_i) is a node of T, and adds c_i to a group key data $GKD = GKD||c_i$.
7. Read a sequence number sq for the destination group DG.
8. Compute $\sigma = \mathrm{Sign}(sk, GI||SN||CS||GKD||\mathsf{SAID}||sq)$.
9. Send $(GI||SN||CS||GKD||\mathsf{SAID}||sq||\sigma)$ to DG.

Procedure of receiver U_i

1. Receive $(GI||SN||CS||GKD||\mathsf{SAID}||sq||\sigma)$.
2. Check sq whether the received message is not a replay attack. If the message with sq was already accepted, U_i stops the subsequent procedure.
3. If $\mathsf{Verif}(pk, GI||SN||CS||GKD||\mathsf{SAID}||sq, \sigma) \neq 1$, U_i stops the subsequent procedure.
4. If U_i has $(I_j, k_j) \in DK_i$ such that $I_j \in CS$,
   (a) compute $\mathsf{mgk} = \mathsf{Unwrap}(k, c_k)$ where $c_k \in GKD$ is the ciphertext corresponding with I_j,
   (b) compute the group session key $\mathrm{gsk} = \mathrm{KDF}(\mathsf{mgk})$, and record $(GI, SN, \mathsf{SAID}, \mathrm{gsk})$.

6 Security Analysis

6.1 Security requirements

We define a formal security model based on BM model [15]. Our security model modifies the definition of *freshness* in BM model to remove *perfect forward secrecy*. This is due to the reason that for IEEE 802.21 applications, reducing the number of multicast communication traffic is an important requirement.

Attack model. Let an adversary $\mathcal{A}$ and the users (including the group manager GM) be probabilistic polynomial-time algorithms. In order to capture multiple sessions, each user U is modeled by an oracle Π_U^s for $s \in \mathbb{N}$. Every session is identified by a unique, publicly-known sid_U^s. Let pid_U^s be a partner id that contains the identities of participating users (including U), and $\mathcal{G}(\Pi_{U_j}^s) = \{\Pi_{U_j}^t \text{where } U_j \in \mathsf{pid}_{U_i}^s \text{and } \mathsf{sid}_{U_i}^s = \mathsf{sid}_{U_j}^t\}$. $\Pi_{U_i}^s$ and $\Pi_{U_j}^t$ are called partner if $\Pi_{U_j}^t \in \mathcal{G}(\Pi_{U_i}^s)$ and $\Pi_{U_i}^s \in \mathcal{G}(\Pi_{U_j}^t)$. $\mathcal{A}$ learns each message to be sent, and it can prevent sending or modifying the message. We assume that receivers always receive the original message sent by the sender, even if $\mathcal{A}$ blocks or modifies it.

$\mathcal{A}$ issues following queries.

- *Initialize*(S): For each user in the set S, a new oracle Π_U^s is initialized and the resulting session id sid is given to $\mathcal{A}$.
- *Invoke*(sid, S'): It assumes that sid is a valid session id and S' is a set of initialized oracles ($S' \subset S$ where S led to the construction of sid). In response, for each $U \in S'$, the oracle Π_U^s turns into the processing stage. If Π_U^s is an initiator of the protocol, Π_U^s outputs the first protocol message.
- *Send*(Π_U^s, m): The message m is sent to Π_U^s. In response, $\mathcal{A}$ receives a processing result of m based on the protocol. The response may be empty, if m is incorrect.
- *Corrupt*(U): In response, $\mathcal{A}$ obtains the long-term key of U, LL_U.
- *AddUser*(U, Λ): In response, a new user U with a long-term key is added to $\mathcal{U}$ where Λ contains the registration information and the long-term key. If the protocol prohibits U from selecting the long-term key, the long-term key in Λ is empty, and in response $\mathcal{A}$ additionally receives U's long-term key.
- *RevealState*(Π_U^s): In response, $\mathcal{A}$ obtains ephemeral secrets stored in state_U^s.
- *RevealKey*(Π_U^s): In response, $\mathcal{A}$ obtains the group session key k_U^s (only if Π_U^s has already accepted).

We say U is corrupted if LL_U is known to $\mathcal{A}$, either via *Corrupt*(U) or *AddUser*(U, Λ); if no such queries have been asked then U is honest.

Definition 5. *(Oracle Freshness) In a session* sid *of* P*, an oracle* Π_U^s *has accepted is fresh if all of the following holds:*

1. *no* $U' \in \mathsf{pid}_U^s$ *has been added by* $\mathcal{A}$ *via corresponding AddUser query,*
2. *no* $U' \in \mathsf{pid}_U^s$ *has been corrupted via corresponding Corrupt query,*
3. *neither* Π_U^s *nor any of its partners is asked for a query RevealState until* Π_U^s *and its partners accept,*
4. *neither* Π_U^s *nor any of its partners is asked for a query RevealKey after having accepted.*

In the original definition in [2], the condition 2. is "*no* $U' \in \mathsf{pid}_U^s$ *is asked for a query Corrupt prior to a query of the form Send*($\Pi_{U_j}^t$, m) *with* $U_j \in \mathsf{pid}_U^s$ *until* Π_U^s *and its partners accept*". It means that Π_U^s is *fresh* even if $U' \in \mathsf{pid}_U^s$ who is a participant of a future session is corrupted, i.e., it represents *perfect forward secrecy*.

In order to provide a formal security proof of our protocol without *perfect forward secrecy*, we modify condition 2 for the weaken $\mathcal{A}$ as descried in Def. 5.

Definition 6. *(Authenticated Key Exchange (AKE) security) Let* P *be a group key distribution protocol. Let* $b \leftarrow_R \{0,1\}$ *and* $\mathcal{A}$ *be an adversary against AKE security of* P*. The attack game* $\mathsf{Game}_{\mathcal{A},\mathsf{P}}^{\mathrm{ake}-b}$ *is defined as follows.*

1. $\mathcal{A}$ *interacts with each oracle using the queries defined in section 6.1.*

2. *$\mathcal{A}$ sends Test query to Π_U^s in arbitrary timing. Π_U^s returns k_U^s if $b = 0$, else U returns a key k_r chosen from the key space uniformly at random.*
3. *$\mathcal{A}$ continues to interact with each oracles using the queries defined in section 6.1.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$ and stops.*

Let Fr *be an event that Π_U^s who receives Test query is still Fresh when $\mathcal{A}$ has been stopped. The advantage of $\mathcal{A}$ is defined as follows:*

$$\mathsf{Adv}_{\mathcal{A},\mathsf{P}}^{\mathrm{ake}-b}(\kappa) = |\Pr[\mathsf{Game}_{\mathcal{A},\mathsf{P}}^{\mathrm{ake}-b}(\kappa) = \mathrm{b} \wedge \mathsf{Fr}] - \tfrac{1}{2}|.$$

We say that a protocol P *is AKE secure if for all probabilistic polynomial-time adversary $\mathcal{A}$,* $\mathsf{Adv}_{\mathcal{A},\mathsf{P}}^{\mathrm{ake}-b}(\kappa)$ *is negligible.*

6.2 Security proof

Theorem 1. *If Σ satisfies EUF-CMA security and $\mathcal{KW}$ satisfies IND-RPA security, the protocol* P *described in section 6 satisfies AKE-security in Def. 6, and*

$$\mathsf{Adv}_{\mathcal{A}_{AKE},\mathsf{P}}^{\mathrm{gk}-b}(\kappa) \leq \frac{(2N-1)\cdot n_s \cdot n_g^*}{2} \cdot \mathsf{Adv}_{\mathcal{A},\mathcal{KW}}^{\mathrm{kw.rpa}}(\kappa) + \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathrm{euf}-cma}(\kappa)$$

where N is the maximum number of users, n_g^* is the number of ciphertexts containing within GKD in $\Pi_{U^*}^s$ who is the receiver of Test query, and n_s is the maximum number of sessions.

Proof of Theorem 1. The security proof is given by the game hopping technique [18]. Let S_i be an event that $b = b'$ and Π_U^{s} who receives Test query is *fresh* at the end of Game *i*.

Game 0: The original attack game of AKE security. Due to Def. 6,

$$\mathsf{Adv}_{\mathcal{A}_{AKE},\mathsf{P}}^{\mathrm{gk}-b}(\kappa) = |\Pr[S_0] - 1/2|. \quad (1)$$

Game 1: Let $\mathrm{L_M}$ be a list of messages issued by GM. In Game 1, each Π_{U}^{s} ignores *Send*(Π_U^{s}, m) if $m \notin \mathrm{L_M}$, and other operations are the same as Game 0.

In the protocol P, the protocol message without the valid signature of GM is dropped by the receivers. The behavior of Π_U^{s} may be different between Game 0 and Game 1, if and only if $\mathcal{A}_{\mathrm{AKE}}$ succeeds the existential forgery of Σ. We assume Σ is EUF-CMA secure, then

$$|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathrm{euf}-cma}(\kappa). \quad (2)$$

Game 2: Let L_U be a list of messages received by the user U. In Game 2, Π_U^{s} ignores *Send*(Π_U^{s}, m) if $m \in \mathrm{L}_U$, and other operations are the same as Game 1.

The message of P contains the sequence number *sq*, and each receiver does not accept the same sequence number. The number of Sent queries is at most polynomial times in

κ since $\mathcal{A}$ is a polynomial-time algorithm. If the space of *sq* is exponentially large in κ, the behaviors of Π_U^s in Game 1 and Game 2 are the same, then

$$\Pr[\mathrm{S}_2] = \Pr[S_1]. \tag{3}$$

Game 3: GM try to guess a session s^* that $\mathcal{A}_{\mathrm{AKE}}$ sends Test query and (I^*, k^*) used in s^*. If it finds that the guess is failed, Game 3 is aborted, and GM decides b' which is the output of Game 3 instead of $\mathcal{A}_{\mathrm{AKE}}$. Game 3 is the same as Game 2 excluding following operations;

- After the *Setup* phase, GM randomly selects a session $s^* \in \{1, \dots, n_s\}$ and a node in the key management tree *T* which has N leaf nodes[1], for guessing the session s^* and the node key k^* used in s^*. Let *Hit* be an event that GM succeeds the guess[2].
- When it finds *Hit* does not occur, Game 3 is aborted and GM decides $b' \leftarrow_{\mathrm{R}} \{0,1\}$ instead of $\mathcal{A}_{\mathrm{AKE}}$.

When *Hit* occurs, Game 3 and Game 2 are the same. When *Hit* does not occur, S_3 occurs at random since GM selects $b' \leftarrow_{\mathrm{R}} \{0,1\}$.

$$\begin{aligned}\Pr[\mathrm{S}_3] &= \Pr[S_3 \wedge Hit] + \Pr[S_3 \wedge \neg Hit] \\ &= \Pr[Hit]\Pr[S_3|Hit] + \Pr[\neg Hit]\Pr[\mathrm{S}_3|\neg Hit] \\ &\geq \frac{1}{(2N-1)n_s}\Pr[S_2] + \frac{1}{2} - \frac{1}{2(2N-1)n_s}\end{aligned} \tag{4}$$

where N is the maximum number of users, n_s is the maximum number of the sessions.

Game 4: In order to estimate $|\Pr[S_3] - 1/2|$, we consider the following hybrid game. In order to replace the reply of Test query with C where $(D_0, D_1, C) \leftarrow \mathrm{LR}_W$, a node key k^* is replaced by *W* using an Wrap_W oracle. Game 4 is the same as Game 3 excluding the following operations;

- For a session s excluding s^*, when the group manager Π_{GM}^s needs a ciphertext with k^* for generating the encrypted group keys GKD, Π_{GM}^s accesses Wrap_W oracle and it receives (D, C). D is regarded as a master group key mgk distributed in s, GKD is generated by C and D with other node keys excluding k^* in *T*, e.g., $\mathrm{Wrap}(k_{i_1}, D), \dots, \mathrm{Wrap}(k_{i_{k-1}}, D), C, \mathrm{Wrap}(k_{i_{k+1}}, D), \dots, \mathrm{Wrap}\left(k_{k_{n_k}}, D\right)$ where n_k is the number of ciphertext contained in GKD.
- For the session s^*, the group manager $\Pi_{GM}^{s^*}$ sends a query to LR_W and receives (D_0, D_1, C^*). D_0 is regarded as a master group key mgk distributed in s^*, and D_1 is regarded as a random key. GKD^* is generated by C^*, D_0, and D_1 with other node keys excluding k^*in *T*, e.g.,

[1] The complete binary tree *T* with *N* leaf nodes has (2*N*-1) nodes.

[2] If the guess is correct, i.e., *Hit* occurs, no U^* assigned (I^*, dk^*) is corrupted at the end of Game 3 since Π_U^* who receives Test query must be *fresh.*

$$\mathrm{Wrap}(k_{i_1}, D_0), \ldots, \mathrm{Wrap}\left(k_{i_{j-1}}, D_0\right), C^*, \mathrm{Wrap}\left(k_{i_{j+1}}, D_1\right), \ldots, \mathrm{Wrap}\left(k_{i_{n_g^*}}, D_1\right)$$

where n_g^* is the number of ciphertext contained in GKD^*

- When $\Pi_{U_j}^{s^*}$ receives Test query, it returns a group session key KDF(D_0).

Let $\mathrm{H_j}$ be a hybrid game that $GKD^* =$ Wrap$(k_{i_1}, D_0), \ldots,$ Wrap$\left(k_{i_{j-1}}, D_0\right)$, Wrap$\left(k_{i_j}, D_0\right)$, Wrap$\left(k_{i_{j+1}}, D_1\right)$, …, Wrap$\left(k_{i_{n_g^*}}, D_1\right)$. Let b_r be the bit of *IND-RPA* game. If $b_r = 0$, Game 3 is the same as $H_{\mathrm{j}+1}$ since $C^*=$ Wrap(W, D_0). If $b_r = 1$, Game 3 is the same as H_j since $C^*=$ Wrap(W, D_1). Let E_i be an event that occurs if $\mathcal{A}_{\mathrm{AKE}}$ outputs 1 in H_i, then $|\Pr[E_{i-1}] - \Pr[E_i]| \leq \mathsf{Adv}_{\mathcal{A},\,\mathcal{KW}}^{\mathrm{kw.rpa}}(\kappa)$ holds. By the hybrid argument, we have

$$\left|\Pr[E_0] - \Pr\left[E_{n_g^*}\right]\right| = \sum\nolimits_{i=1}^{n_{g^*}} |\Pr[E_{i-1}] - \Pr[E_i]| \leq n_g^* \cdot \mathsf{Adv}_{\mathcal{A},\,\mathcal{KW}}^{\mathrm{kw.rpa}}(\kappa).$$

Accordingly, H_0 is the same as Game 3 when $b = 1$, and $H_{n_g^*}$ is the same as Game 3 when $b = 0$;

$$\begin{aligned}
\Pr[\mathrm{S}_3] - 1/2 &= |\Pr[\mathcal{A}_{\mathrm{AKE}} \to 1 \text{ in Game 3 } \wedge \mathrm{b} = 1] \\
&\qquad + \Pr[\mathcal{A}_{\mathrm{AKE}} \to 0 \text{ in Game 3 } \wedge \mathrm{b} = 0] - 1/2| \\
&= \frac{1}{2} |\Pr[\mathcal{A}_{\mathrm{AKE}} \to 1 \text{ in Game 3 } |\mathrm{b} = 1] \\
&\qquad - (1 - \Pr[\mathcal{A}_{\mathrm{AKE}} \to 1 \text{ in Game 3 } |\mathrm{b} = 0]) - 1| \\
&= \tfrac{1}{2} \left|\Pr[E_0] - \Pr\left[E_{n_g^*}\right].\right|
\end{aligned} \tag{5}$$

According to Eq. (1), (2), (3), (4), and (5),

$$\begin{aligned}
&\mathsf{Adv}_{\mathcal{A}_{AKE},\mathrm{P}}^{\mathrm{gk}-b}(\kappa) = \left|\Pr[S_0] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathrm{S}_1] + |\Pr[\mathrm{S}_1] - \Pr[\mathrm{S}_0]| - \frac{1}{2}\right| \leq \left|\Pr[S_1] + \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathrm{euf}-cma}(\kappa) - \frac{1}{2}\right| \\
&= \left|\Pr[\mathrm{S}_2] + \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathrm{euf}-cma}(\kappa) - \frac{1}{2}\right| \\
&= \left|(2N - 1) \cdot n_{\mathrm{s}} \left(\Pr[\mathrm{S}_3] - \frac{1}{2}\right) + \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathrm{euf}-cma}(\kappa)\right| \\
&\leq \frac{(2N - 1) \cdot \mathrm{n_s} \cdot \mathrm{n_g^*}}{2} \cdot \mathsf{Adv}_{\mathcal{A},\,\mathcal{KW}}^{\mathrm{kw.rpa}}(\kappa) + \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathrm{euf}-cma}(\kappa).
\end{aligned}$$

7 Prototyping

We implemented the group key distribution protocol as described in section 5 and measured the processing time of group manager (GM) and receivers. In real systems, the receivers may have memory constraints. For such system, the code footprint size is also important. Therefore, we also measure the footprint size of the receivers. Table 1 shows the benchmark of the computing machine used for GM and receivers.

Table 1. Computing Machine Specification

	GM	Receivers
CPU	Core i5-4310M, 2.7 GHz	ARM11176JZF-S, 700 MHz
RAM	4GB	512 MB
OS	Ubuntu 14.04.4	Raspbian

We considered the number of receivers is 1024, with threshold of fragmentation as 32. Table 2 shows processing time that GM takes to generate the protocol messages and the receiver takes to process them. During processing cycle, signing time and verification time of ECDSA are dominant, when ECDSA is attached to each messages. Table 3 shows the size of the protocol messages. The processing time and message size depend on the selection of group members.

Table 2. Processing times

	GM		Receiver	
	Average [msec]	Max [msec]	Average [msec]	Max [msec]
Best case	4.71	4.74	265.15	303.59
Worst case	83.80	85.01	4253.91	4276.05

Table 3. Protocol message size

	Message size[bytes]
Best case	272
Worst case	18336

In a best case scenario, $GroupKeyData$ contains only one ciphertext and hence GM sends only one message. In worst case scenario, GM sends 512 ciphertexts, where GM issues 16 messages with $GroupKeyData$ which contains 32 ciphertexts (given the threshold of fragmentation is 32). In our implementation, when the receiver receives a message, it first verifies an ECDSA signature in the message. Therefore in worst case scenario, receiver verifies 16 messages and hence the processing time increases. On the decryption side though the receiver needs to decrypt only one message that carries the complete subtree covering the receiver in order to extract the group key, and other 15 messages can be ignored after verification. So even in worst case, there is a significant advantage in terms of overall processing time.

Table 4 shows foot print size of the receivers.

Table 4. Footprint size of Receivers

	size [Byte]
$heap$	55264
$stack$	12200
$text$	172257
$data$	1268
bss	119900

Total	360889

We measured the memory usage occupied on the virtual memory space. The *text* segment corresponds to the code size. The *data* and *bss* segments include pre-defined variables (*data* has initialized data, while *bss* is uninitialized). The *stack* area is used for storing temporal variables on the program that is executed. The *heap* area is managed by malloc()-like functions. Each size of *text*, *data*, and *b*ss segments are fixed in every program executions. These values are measured by the size of the command. In this early prototype implementation, we focus on reducing the heap size for to simplify memory management simplification. The maximum sizes of *stack* or *heap* areas are measured by valgrind^{TM} [19].

8 Conclusion

We introduced a secure multicast-based group key management and key distribution protocol that is recently standardized in IEEE 802.21. Although it is based on the concept of logical key hierarchy, a method has been specified on how the 'Complete Subtree' can be used to optimize the number of encryptions for each group key distribution. A data format called 'Group Data Block' has been used for encoding the 'Complete Subtree' and other data associated with it. To support the practical applications, the standard assumes an architecture whereby a group manager is responsible for distributing the group key. The group key distribution protocol uses a deterministic symmetric key wrapping scheme and a digital signature scheme. A formal security analysis and corresponding proof have been performed based on Bresson and Manulis model. While additional work is required, an early prototype implementation results with 1024 nodes and tree depth of 7 show that the scheme is realizable in memory constrained devices. It provides an easy way to securely add and remove group members.

References

1. IEEE Standard for Local and metropolitan area networks- Part 21: Media Independent Handover Services- IEEE Std 802.21™-2008, January 2009.
2. IEEE Standard for Local and metropolitan area networks- Part 21: Media Independent Handover; Amendment 1: Security Extensions to Media Independent Handover Services and Protocol, May 2012.
3. IEEE Standard for Local and metropolitan area networks- Part 21: Media Independent Handover; Amendment 4: Multicast Group Management, July 2015.
4. D. Wallner, E. Harder, and R. Agee Key Management for Multicast: Issues and Architectures Request for Comments: 2627 June 1999.
5. C. K. Wong, M. Gouda, and S. S. Lam, "Secure Group Communications Using Key Graphs," IEEE/ACM Transactions on Networking, vol. 8, No. 1, pp. 16-30, 2000.
6. ISO/IEC 11770-5 Information technology – Security techniques - Key management – Part 5: Group key management, 2011.
7. A. Fiat and M. Naor, Broadcast Encryption, Advances in Cryptology - CRYPTO'93, LNCS 773, Springer, 1994, pp. 480-491.

8. B. Weis, S. Rowles, and T. Hardjono The Group Domain of Interpretation IETF, Request for Comments 6407, October 2011
9. IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2015.
10. D. Naor, M. Naor, and J. Lotspiech, Revocation and tracing schemes for stateless receivers. In Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings, pages 41–62, 2001.
11. W. Diffie, P. C. van Oorschot, and M. J. Wiener. A key exchanges. Des. Codes Cryptography, 2(2):107–125, 1992.
12. M. Burmester, On the risk of opening distributed keys. In Advances in Cryptology CRYPTO ’94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings, pages 308–317, 1994
13. Y. Kim, A. Perrig, and G. Tsudik, Simple and fault-tolerant key agreement for dynamic collaborative groups. In CCS 2000, Proceedings of the 7th ACM Conference on Computer and Communications Security, Athens, Greece, November 1-4, 2000., pages 235–244, 2000.
14. C. G. Gu¨nther. An identity-based key-exchange protocol. In Advances in Cryptology EUROCRYPT ’89, Workshop on the Theory and Application of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, Proceedings, pages 29–37, 1989.
15. T. Brecher, E. Bresson, and M. Manulis, Fully robust tree-diffie-hellman group key exchange. In Cryptology and Network Security, 8th International Conference, CANS 2009, Kanazawa, Japan, December 12-14, 2009. Proceedings, pages 478–497, 2009.
16. R. Gennaro and S. Halevi, More on key wrapping. In Selected Areas in Cryptography 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers, pages 53–70, 2009.
17. Burton H. Bloom, Space/time trade-offs in hash coding with allowable errors, Communications of the ACM, vol. 13, Issue 7, July 1970.
18. V. Shoup, Sequences of games: a tool for taming complexity in security proofs, IACR Cryptology ePrint Archive, 2004:332, 2004.
19. Valgrind, http://valgrind.org/