Efficient Group Key Management in Wireless LANs

Celia Li and Uyen Trang Nguyen
Department of Computer Science and Engineering
York University, Toronto, Ontario M3J 1P3, Canada
Email: {cli, utn}@cs.yorku.ca

***Abstract*—Group key management (GKM) refers to the actions taken to update and distribute the group key upon members joining and leaving a multicast/broadcast group. The GKM scheme defined in the IEEE 802.11 WLAN standards is not efficient because the re-keying latency grows linearly with the number of clients connected to an access point. In this paper, we apply the logical key hierarchy (LKH) and one-way function tree (OFT) algorithms to GKM in WLANs to improve its performance. Our numerical analyses show that the LKH and OFT algorithms reduce the re-keying latency of GKM in WLANs to logarithmic time.**

***Index Terms*—Group key management; logical key hierarchy; one-way function tree.**

I. Introduction

Multicast is a form of communication that delivers information from a source to a set of destinations simultaneously in an efficient manner; the messages are delivered over each link of the network only once and only duplicated at branch points, where the links to the destinations split. Important applications of multicast include distribution of financial data, billing records, software, newspapers, pay-per-view movies; audio/video conference; distance learning; and distributed online games. In this paper, we address one of the most essential issues of multicast in WLANs − security. In particular, we focus on the issue of group key management for multicast in WLANs.

In order to ensure that only authorized users can access the multicast data, the data are encrypted using a cryptographic key known as the group key. The group key is known only to authenticated and authorized members of a multicast group. Every time a membership change occurs, the group key must be changed to ensure backward and forward secrecy. Backwards secrecy guarantees that a new user joining the multicast group does not have access to any old keys. This ensures that a member cannot decrypt messages sent before it joins the group. Forward secrecy requires that a member leaving the group does not have access to any future keys. This ensures that a member cannot decrypt future messages after it leaves the group. Group key management refers to the actions taken to update and distribute the group key upon members joining and leaving a multicast group.

In IEEE 802.11 WLANs, an access point (AP) shares a pairwise key with each mobile device it is connected to. In addition, the AP shares a group key with all the trusted mobile devices associated with the AP so that it can send multicast and broadcast data to this trusted group.

When a client joins or leaves the WLAN, the AP has to update the group key to ensure backward and forward secrecy. The AP generates a new group key, encrypts it using the pairwise key the AP shares with each trusted member, and sends the encrypted information to the members one by one. It can be seen that the communication cost of a group key update is $O(n)$ where n is the number of associated members. As the group size becomes large, the re-keying latency becomes unacceptably inefficient, and thus not scalable.

The objective of our work is to improve the latency of group key updates in WLANs. In particular, we apply the logical key hierarchy (LKH) [1] and one-way function tree (OFT) [6] algorithms to GKM in WLANs. We first show how to incorporate the LKH and OFT algorithms into the key management protocol defined in IEEE 802.11. We then compare the group key update latencies of LKH, OFT and IEEE 802.11 GKM via numerical analyses. Our analyses show that the LKH and OFT algorithms reduce the re-keying latency to logarithmic time.

Note that our work in this paper is GKM at the medium access control (MAC) layer as considered in IEEE 802.11i standards. The AP of a basic service set (BSS) and the mobile devices connected to the AP in that BSS are considered a group. This is not to be confused with GKM at the application layer in which all the members of a group run/use the same application and may be located over different geographical areas.

The remainder of this paper is organized as follows. In Section II, we provide background information and related work on group key management. Section III provides an overview of LKH and OFT algorithms. LKH and OFT are adapted to 802.11 in Section IV. We analyze the performance of each group key management in Section V. Section V-C6 discusses the finding results. Section VI summarizes the paper and outlines our future work.

II. Related Work

In this section, we summarize existing work on GKM in both wired and wireless networks. GKM can be broadly categorized into three approaches: centralized, decentralized and contributory key updating protocols.

The *centralized approach* [1], [5], [6], [7], [8] relies on a *central controller* − a trusted third party − that generates and distributes the group key. Representative algorithms using this approach are the logical key hierarchy (LKH) [1] and one-way function tree (OFT) [6] schemes. LKH and OFT use a

hierarchical key structure called *logical key tree* to make group key distributions and updates scalable. This is in contrast to the current GKM of IEEE 802.11 MAC, which uses a flat structure and is thus not scalable when the number of group members grows.

In the *decentralized approach* [9], [10], a multicast group is organized into smaller subgroups with multiple subgroup controllers. Each subgroup has its own subgroup key and is managed by a local controller. The local controller is in charge of key computation and distribution within its subgroup. Different subgroups may use different GKM protocols. A membership change impacts only the subgroup of the member: only the subgroup key needs to be updated, independently of the keys of the other subgroups. The decentralized architecture ensures scalability for GKM in very large scale networks where the members of a group may be distributed over different and vast geographical areas.

In contrast to the centralized and decentralized approaches, *contributory GKM* [11], [12], [13] requires each group member to contribute an equal share to the common group key (which is then computed as a function of all members' contributions). Most contributory protocols do not require pairwise secure communication channels between group members. This is a desirable feature for wireless ad-hoc and sensor networks where communication channels are vulnerable to eavesdropping and attacks, and previously established shared keys may not be readily available. In a contributory key management scheme, there is no explicit central controller, and keys are generated collaboratively by one or multiple group members. Many GKM algorithms proposed for wireless ad-hoc and sensor networks fall into this category.

Among the three approaches, the centralized approach is the most efficient and cost-effective for GKM at the MAC layer in a BSS of a WLAN for the following reasons. First, there already exists a central controller that can generate and distribute the group key to the members, which is the access point (AP) of a basic service set (BSS). Second, the members of the group – the mobile devices connected to the AP – are physically located close to the central controller. Third, there already exists a secure communication channel between the AP and each mobile device, which is provided by a shared key between the AP and the device called a *pairwise transient key* (PTK) in IEEE 802.11 standards.

In the next sections, we present an overview of the LKH and OFT algorithms and discuss how to adapt them to GKM in WLANs.

III. Overview of LKH and OFT Algorithms

Both LKH and OFT algorithms use a hierarchical key structure called *logical key tree* to ensure scalable key updates. LKH is a top-down method in the sense that it "pushes" new group keys from the root down to the leaves of the key tree. In contrast, OFT is a bottom-up method because new group keys are derived from the leaves going up to the root of the key tree.

A. Logical Key Tree

The key tree is a logical data structure used in hierarchical GKM schemes. (This is not to be confused with the physical multicast routing tree of the same group, or the recovery tree used in reliable multicast for retransmissions of lost/damaged data packets.) The logical key tree provides scalable computation, maintenance and updates of the group key.

Consider a group with six members C_1, ..., C_6. A logical key tree for this group is shown in Fig. 1. (To simplify the discussions, we assume binary trees in this paper, although trees of higher degrees can be used with LKH and OFT.) Let K_i denote the content of a node i in the key tree. Each leaf node i is associated with a group member, and contains the member's individual key denoted by κ_i.

$$K_i = \kappa_i$$

In the logical key tree shown in Fig. 1, the leaf nodes 6, 7, ..., 11 contain the individual keys $\kappa_6, \kappa_7, \ldots, \kappa_{11}$ of the six group members $C_1, C_2, \ldots, C_6$.

Non-leaf node are not associated with any members, but are virtual nodes. The content stored in each non-leaf node is a key called *intermediate key*. The content of the root, node 1, is the *group key* K_1, which is used by the source of the group to encrypt data packets before sending them to the group members, and by the group members to decrypt the packets.

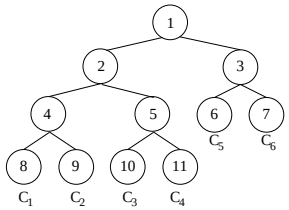


Fig. 1. A logical key tree

The group key needs to be updated when a node joins or leaves the group to ensure forward and backward secrecy. It also needs to be refreshed periodically even when there are no membership changes to prevent an attacker from gathering sufficient time or resources to break the key. Following are the procedures for updating the group key when a member joins or leaves the group using the LKH and OFT algorithms. To simplify the discussions, we assume that logical key trees are binary trees, and the leaf nodes are added to the tree from left to right and top to bottom. The notation $\{X\}_Y$ denotes the encryption of content X using key Y.

B. LKH Operations

The LKH method [1] is a tree-based GKM scheme using symmetric-key cryptography. In LKH method, the root node of the key tree stores the group key, which is shared by all members in the group. Each group member is associated with a leaf node, which contains an individual key of each member. An individual key is shared only between the member owning the key and the group server, and is used for pairwise confidential communication between them. The non-leaf nodes,

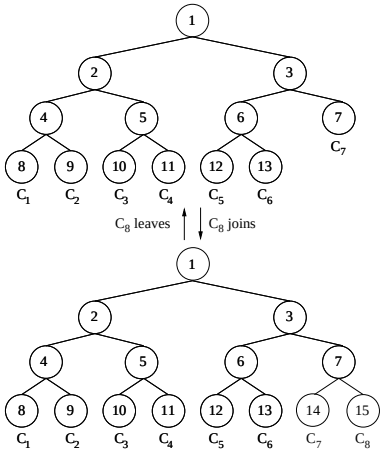


Fig. 2. Client C_8 joins/leaves the group

except the root node, are intermediate keys that are used to encrypt other intermediate or group keys during a re-keying process and not data. Each member of the group stores the keys along the path from the root to the leaf node assigned to that member (i.e., $\lceil \log_2 n \rceil$ keys, where n is the number of group members).

1) Join Operation: After granting a join request from a client C, the group server creates a new node for client C. The server finds an existing node (called the joining point) in the key tree and attaches the new node to the joining point as a new leaf node. Consider an example illustrated by Fig. 2 in which client C_8 joins (leaves) the group. The joining point is node 15.

To prevent the joining client from accessing past messages, all keys along the path from the joining point to the root node need to be changed. After generating new keys for these nodes, the server needs to securely distribute them to the new member as well as the current members.

For the new member C, the server encrypts all the non-leaf node keys on the path from the root to C using C's individual key and sends the encrypted message to C. In the above example, the server sends the following message to C_8:

$s \rightarrow \{C_8\} : \{K_1', K_3', K_7'\}_{\kappa_{15}}$

C will decrypt the message and obtain those keys.

For the current members, each new key K_i' is encrypted using the current key K_i occupying the same node i. In the above example, the server will send the following messages:

$s \rightarrow \{C_1, C_2, \ldots, C_7\} : \{K_1'\}_{K_1}$

$s \rightarrow \{C_5, C_6, C_7\} : \{K_3'\}_{K_3}$

$s \rightarrow \{C_7\} : \{K_7'\}_{K_7}$

Note that the server may combine multiple messages into a physical packet before sending to the members in order to save network bandwidth. In Section IV-A, we will discuss how this is done when the LKH algorithm is applied to GKM in WLANs.

2) Leave Operation: After granting a leave request from client C, the group server updates the key tree by deleting the leaf node that contains client C's individual key from the key tree. The parent of the leaf node to be deleted is called the leaving point. To prevent the leaving client from accessing future group data, all keys along the path from the leaving point to the root node need to be changed. After generating new keys for these nodes, the server needs to securely distribute them to the remaining members. Similarly to the join operation, each new key K_i' is encrypted using the current key K_i occupying the same node i. Given the example shown in Fig. 2, when client C_8 leaves the group, the server will send the following re-keying messages to the remaining members.

$s \rightarrow \{C_1, C_2, C_3, C_4\} : \{K_1'\}_{K_2}$

$s \rightarrow \{C_5, C_6, C_7\} : \{K_1'\}_{K_3'}$

$s \rightarrow \{C_5, C_6\} : \{K_3'\}_{K_6}$

$s \rightarrow \{C_7\} : \{K_3'\}_{\kappa_7}$

C. OFT Operations

In OFT [6], the keys are computed up the tree, from the leaves to the root. The group server maintains a balanced binary key tree for the group. Each node i associates three types of keys used in OFT: node secret K_i, node key $g(K_i)$ and blinded node key $f(K_i)$. Blinded node key $f(K_i)$ is defined as a one-way function of the node secret K_i. A pseudo-random function f is used to compute blind key. (Pseudo-random functions are used to generate random numbers.) It is blinded in the sense that a computationally limited adversary can know $f(K_i)$, yet cannot find K_i. Another pseudo-random functions g is used to compute node key $g(K_i)$ for each node and to encrypt other keys.

The OFT key tree is a particular type of binary tree in which each intermediate node has exactly two children. Each leaf of the tree is associated with a group member, and the node secret of the root is the group key. For any intermediate node i in the key tree, the node key K_i of node i is defined by $K_i = f(K_L) \oplus f(K_R)$., where L and R denote the left and right children of node i, respectively,

Each group member maintains the node secret of the leaf with which it is associated, and a list of blinded node secrets for all of the siblings of the nodes along the path from itself to the root. This information enables the member to compute the node secrets along its path to the root, including the root key. This information also enables the member to compute the node keys along this path. If one of the node secret changes and the member is told the new value, then it can recompute the node secret on its path to the root and find the new group key.

1) Join Operation: When a new member joins a group, an existing leaf node i is split, the member associated with i is associated with $left(i)$, and the new member is associated with $right(i)$. Both members are given new node secret, which will affect all node secrets along their path to the root. The new blinded node secrets that have changed are broadcast to the appropriate subgroup members. More specifically, if i is any nonroot node along the affected path and, if j is the sibling of i, then the group server broadcasts the blinded new node secret $f(K_i)$ of i encrypted with the node key $g(K_j)$ of j. Doing so enables all descendants of j to learn the new node secret

$f(K_i)$. In addition, the new member is given a set of blinded node secrets in a unicast transmission using the external secure channel. The key server and all members individually compute the new group key.

For example, as shown in Figure 2, for client C_8 to join the group, group server needs to send the following four rekey messages.

$s \rightarrow \{C_1, C_2, C_3, C_4\} : \{f(K_3')\}_{g_{K_2}}$
$s \rightarrow \{C_7\} : f(\kappa_1' 4, f(\kappa_1 5))_{g_{\kappa_{14}}}$
$s \rightarrow \{C_5, C_6\} : \{f(K_7')\}_{g_{K_6}}$
$s \rightarrow \{C_8\} : f(\kappa_{14}', f(K_6), f(K_2))_{g_{\kappa_{15}}}$

Upon receiving the above re-keying messages, each member can compute the group key as follows.

Client C_1, C_2, C_3 and C_4 can compute the group key as $K_1' = f(K_2) \oplus f(K_3')$. Note that clients C_1, C_2, C_3 and C_4 can compute $f(K_2)$ as they all know the node key K_2.

Client C_5 and C_6 can compute the group key in the following two steps as $K_3' = f(K_6) \oplus f(K_7')$, $K_1' = f(K_2) \oplus f(K_3')$. Since clients C_5 and C_6 know the node secret K_6, they can compute the group key K_1' after receiving the updated blinded node secret $f(K_7')$.

Client C_7 and C_8 can compute the new group key as $K_7' = f(\kappa_{14}) \oplus f(\kappa_{15})$, $K_3' = f(K_6) \oplus f(K_7')$,$K_1' = f(K_2) \oplus f(K_3')$.

2) Leave Operation: If a member associated with a leaf node i leave the group, j is the sibling of i. Let p be the parent of i. If j is a leaf node, then the member assigned to j is reassigned to p and given a new node secret K_p'. All node keys along the path from the changed leaf node key to the root are affected. The group server broadcasts all new blinded node secrets that have changed to its group members.

For instance, client C_8 in Figure 2 leaves the group, and the the group server sends the following rekeying messages to the whole group to update the keys:

$s \rightarrow \{C_1, C_2, C_3, C_4\} : \{f(K_3')\}_{g_{K_2}}$
$s \rightarrow \{C_5, C_6\} : \{f(\kappa_7')\}_{g_{K_6}}$
$s \rightarrow \{C_7\} : \kappa_{7_{g(\kappa_7)}}'$

By receving the above rekeying messages, each member can compute the group key as follows.

Client C_1, C_2, C_3 and C_4 can compute the group key as $K_1' = f(K_2) \oplus f(K_3')$. Note client C_1, C_2, C_3 and C_4 can compute $f(K_2)$ as they all know the node key K_2.

Client C_5 and C_6 can compute the group key in the following two steps as $K_3' = f(K_6) \oplus f(\kappa_7')$, $K_1' = f(K_2) \oplus f(K_3')$. Since client C_5 and C_6 know the node secret K_6, they can compute the group key K_1' after receiving the new blinded node secret $f(\kappa_7')$.

Client C_7 can compute the new group key as $K_3' = f(K_6) \oplus f(\kappa_7')$, $K_1' = f(K_2) \oplus f(K_3')$.

IV. Implementing LKH and OFT in 802.11 WLANs

We apply the LKH and OFT algorithms described above to GKM at the MAC layer in WLANs. The group server is now the access point (AP) and the group members are the clients associated with the BSS under the control of the AP. The AP maintains and update the logical key tree. As mentioned earlier, the AP may combine several re-keying messages for different members into one packet to save network bandwidth. In this section, we present the message structure used by the LKH and OFT algorithms for GKM at the MAC layer in WLANs. The design goal of the message format is to minimize the communication cost between the AP and the members during the re-keying process.

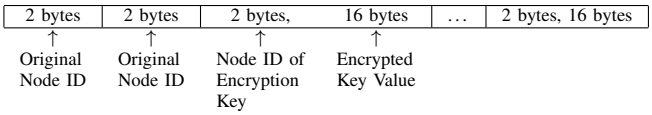
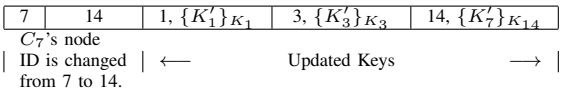
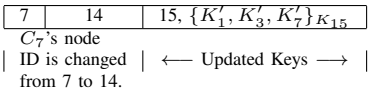


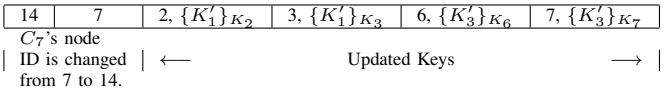
Fig. 3. Message format



(a) Broadcast message for client C_1 to C_7 after client C_8 joins the group.



(b) Unicast message for client C_8 after client C_8 joins the group.



(c) Broadcast message after client C_8 leaves the group.

Fig. 4. LKH message example after client C_8 joins/leaves the group

A. LKH Message Structure

Each updated key is sent out with its corresponding node ID in the key tree. The key size is 16 bytes (128 bits) as we choose AES as the encryption algorithm. The first four bytes of the message indicates the node in the key tree whose position has been changed due to a client joining/leaving the group. The first two of these four bytes indicate the original node ID (original position in the key tree), and the other two bytes indicate the new node ID (new position in the key tree) of an existing member. The remaining bytes are divided into groups of 18 bytes each. Each 18-byte group contains a 2-byte node ID and 16-byte encrypted key value. The 2-byte node ID identifies the node storing the key k that is used to encrypt one or more new keys. The 16-byte encrypted key value is obtained from encrypting one or more newly generated keys using key k.

Consider the example in Figure 2. When client C_8 joins the group, the individual key of client C_7 will be moved from node 7 to node 14. In the key updating message, the first two bytes will record the original position "7", and the next two bytes will store the new position, "14". In the next 18 bytes, the first two bytes store the node ID of the encryption key followed by the 16-byte encrypted key value. For example, in Figure 4, "$1, \{K_1'\}_{K_1}$" denotes that the newly generated key K_1' is encrypted with key K_1. The encryption key K_1 is located at node 1 in the key tree. Similarly, "$14, \{K_7'\}_{\kappa_1}$" denotes that the newly generated key K_7' is encrypted with key κ_{14}. The encryption key κ_{14} is located at node 14 in the key

7	14	2, $\{f(K_3')\}_{g(K_2)}$	6, $\{f(K_7')\}_{g(K_6)}$	14, $\{\kappa_{14}', f(\kappa_{15})\}_{g(K_{14})}$

C_7's node ID is changed from 7 to 14. ⟵ Updated Keys ⟶

(a) Broadcast message for client C_1 to C_7 after client C_8 joins the group.

7	14	15, $\{f(\kappa_{14}'), f(K_6), f(K_2)\}_{g(\kappa_{15})}$

C_7's node ID is changed from 7 to 14. ⟵ Updated Keys ⟶

(b) Unicast message for client C_8 after client C_8 joins the group.

14	7	2, $\{f(K_3')\}_{g(K_2)}$	6, $\{f(K_7')\}_{g(K_6)}$	7, $\{\kappa_7'\}_{g(\kappa_7)}$

C_7's node ID is changed from 7 to 14. ⟵ Updated Keys ⟶

(c) Broadcast message after client C_8 leaves the group.

Fig. 5. OFT message example after client C_8 joins/leaves the group

tree. Figure 4 shows the contents of LKH re-keying messages when client C_8 joins/leaves the group.

B. OFT Message Structure

To apply the OFT algorithm to GKM in WLANs, we use the same message structure as shown in Figure 3 and described in Section IV-A.

In the example in Figure 2, when client C_8 joins the group, the individual key of client C_7 will be moved from node 7 to node 14. In the re-keying message, the first two bytes will record the original position "7", and the next two bytes will store the new position, "14". In the next 18 bytes, the first two bytes store the node ID of the encryption key followed by the 16-byte encrypted key value. For example, in Figure 5, $2, \{f(K_3')\}_{g(K_2)}$ indicates that the newly generated blinded node secret $f(K_3')$ is encrypted with node key $g(K_2)$, where $g(K_2)$ is the result by applying hashing function g to the node secret K_2 of node 2. Similarly, $14, \{\kappa_{14}', f(\kappa_{15})\}_{g(\kappa_{14})}$ indicates that the newly generated node secret κ_{14}' is encrypted with node key $g(\kappa_{14})$, where $g(\kappa_{14})$ results from applying hashing function g to node secret κ_{14} stored in node 14. Figure 5 shows the contents of OFT re-keying messages when client C_8 joins/leaves the group.

V. Performance Evaluation

In this section, we analyze and compare the re-keying latencies of LKH and OFT with that of the IEEE 802.11 GKM scheme (which will be denoted by 802.11 in the remainder of this section).

A. Computational Complexity

Let n be the number of members in the multicast group, and assume a proper binary key tree. Among the three algorithms to be compared, 802.11, LKH and OFT, only OFT uses hash functions. The number of hashing operations performed by OFT is provided in Table I. Table II summarizes the computational complexity of the 802.11, LKH and OFT algorithms, which shows that 802.11 takes $O(n)$ time while LKH and OFT both take $O(\log n)$ time.

TABLE I
Number of hash operations in OFT

Description		$f(K)$	$g(K)$
Join	Client	1	1
	Access Point	$2log_2n$	log_2n+1
	Total Hashing	$2log_2n+1$	log_2n+2
Leave	Client	1	1
	Access Point	log_2n	log_2n+1
	Total Hashing	log_2n+1	log_2n+2

B. Communication Complexity

The communication complexity is measured in terms of the number of keys to be broadcast and unicast during a re-keying operation. Table III summarizes the communication complexity of the join and leave operations incurred by the 802.11, LKH and OFT algorithms. Following is a brief description of the communication complexity given in Table III.

In the 802.11 algorithm, every time the AP updates group key, it has to send (unicast) the new key to the members one by one, hence $O(n)$ time for both join and leave operations.

In the LKH algorithm,

− when a client joins the group, the AP broadcasts a message containing $\log_2 n$ updated keys to the group. The AP also unicasts a message containing $\log_2 n$ updated keys to the new member.

− when a member leaves the group, the AP broadcasts a message containing $2\log_2 n$ updated keys to the group.

In the OFT algorithm,

− when a client joins the group, the AP broadcasts a message containing $\log_2 n$ new blinded node secrets to the group. A new key assigned to the sibling node of the new member is also part of the broadcast message, which contains a total of $\log_2 n + 1$ keys. In addition, the AP unicasts a message containing $\log_2 n$ new blinded node secrets to the new member.

− when a member leaves the group, the AP broadcasts a message containing $\log_2 n$ new blinded node secrets to the group. A new key assigned to the modified leaf node is

TABLE II
Computational complexity

Operation		802.11	LKH	OFT
Join	Encryption	n	log_2n+1	log_2n+1
	Decryption	1	log_2n	log_2n
	Hashing			$3log_2n+2$
Leave	Encryption	n	$2log_2n$	log_2n+1
	Decryption	1	log_2n	1
	Hashing			$2log_2n+2$

TABLE III
Communication complexity

Operation/ Algorithm		Unicast (number of keys)	Broadcast (number of keys)
Join	802.11	n	
	LKH	log_2n	log_2n
	OFT	log_2n	log_2n+1
Leave	802.11	n	
	LKH		$2log_2n$
	OFT		log_2n+1

also part of the broadcast message, which contains a total of $\log_2 n + 1$ keys.

C. Numerical Analysis

In this section, we derive the group key update latency functions for the 802.11, LKH and OFT algorithms. The re-keying latency is defined as the interval starting when the AP receives a join/leave request from a member and initiates the re-keying process and ending when all members receive the new group key.

1) System Model: Given the message format in Fig. 3, the size of an MSDU is determined by the number of keys carried by the MSDU, as follows: $MSDU(K) = 4 + 18K$

The 802.11, 802.11a and 802.11b standards specify a maximum MSDU (MAC Service Data Unit) size of 2304 bytes [14], [15], [16]. Thus one message can store at most 127 updated keys.

The MAC 802.11 protocol with Distributed Coordination Function (DCF) is chosen as the medium access control protocol. Multicast and broadcast transmissions use CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). Unicast transmissions also use RTS/CTS/DATA/ACK exchanges in addition to CSMA/CA. We consider two spread spectrum technologies: direct sequence spread spectrum (DSSS) and orthogonal frequency division multiplexing (OFDM).

We also make the following assumptions:

- No bit error.
- No losses due to collision.
- No packet loss due to buffer overflow at the receiver node.
- Sufficient packets to be send by sending node.
- No fragmentation at the MAC layer.
- No management frame considered (e.g. beacon, association frames).

The contention window size (cw) does not increase exponentially since we assume that there are no collisions. Thus, cw is always equal to the minimum contention window size (cw_{min}) whose value depends on the spread spectrum technology. The backoff (BO) time is selected randomly following a uniform distribution from (0, cw_{min}) given the expected value $cw_{min} = 2$.

The re-keying latency upon a join or leave operation consists of two costs: communication cost and computation cost. We discuss the calculations of these two costs next.

2) Communication Cost: Fig. 6(a) shows the sequence of messages exchanged to complete a unicast transmission at the MAC layer. According to the diagram, the latency $T_u(K)$ incurred by a successful unicast transmission is as follows:
$T_u(K) = T_{BO} + T_{DIFS} + T_{RTS} + T_{CTS} + 3T_{SIFS} + T_{Data}(K) + T_{ACK}$
Note that the unicast transmission latency T_u is a function of the message size, and thus the number of keys K stored in the message.

Following is the latency $T_b(K)$ incurred by a successful broadcast transmission, according to the diagram illustrating the timeline of a broadcast transmission at the MAC layer in Fig. 6(b).

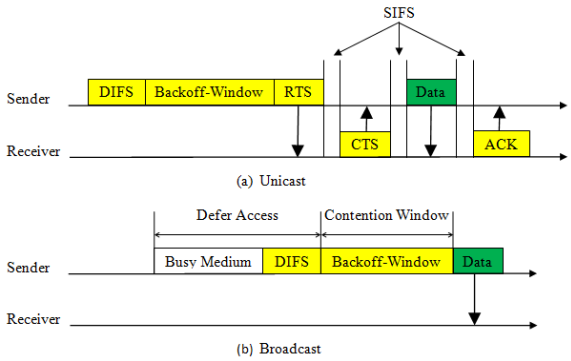


Fig. 6. Unicast and broadcast message exchanges at the MAC layer

$T_b(K) = T_{BO} + T_{DIFS} + T_{Data}(K)$

3) Computation Cost: Let E, D and H denote the latency of an encryption, decryption and hashing operation, respectively. By referring to Table II, we obtain the computation costs of the join and leave operations in the three algorithms as shown in Table IV (denoted by "Encryption", "Decryption", and "Hashing" rows).

TABLE IV
REKEYING LATENCY

Operation		802.11	LKH	OFT
Join	Unicast	$T_u(n)$	$T_u(log_2 n)$	$T_u(log_2 n)$
	Broadcast		$T_b(log_2 n)$	$T_b(log_2 n + 1)$
	Encryption	$n \times E$	$(log_2 n + 1) \times E$	$(log_2 n + 1) \times E$
	Decryption	$1 \times D$	$log_2 n \times D$	$log_2 n \times D$
	Hashing			$(3log_2 n + 2) \times H$
Leave	Unicast	$T_u(n)$		
	Broadcast		$T_b(2log_2 n)$	$T_b(log_2 n + 1)$
	Encryption	$n \times E$	$2log_2 n \times E$	$(log_2 n + 1) \times E$
	Decryption	$1 \times D$	$log_2 n \times D$	$1 \times D$
	Hashing			$(2log_2 n + 2) \times H$

4) Total Re-keying Latency: Table IV also shows the communications costs incurred by the three algorithms (denoted by "Unicast" and "Broadcast" rows). The arguments of the T_u and T_b functions are the numbers of keys stored in the messages as listed in Table III.

The re-keying latency – combination of computation and communication costs – incurred by a join operation in each of the three algorithms is as follows:
$J_{802.11} = T_u(n) + n \times E + 1 \times D$
$J_{LKH} = T_u(log_2 n) + T_b(log_2 n) + (log_2 n + 1) \times E + log_2 n \times D$
$J_{OFT} = T_u(log_2 n) + T_b(log_2 n + 1) + (log_2 n + 1) \times E + log_2 n \times D + (3log_2 n + 2) \times H$

The re-keying latencies incurred by a leave operation are as follows:
$L_{802.11} = T_u(n) + n \times E + 1 \times D$
$L_{LKH} = T_b(2log_2 n) + 2log_2 n \times E + log_2 n \times D$
$L_{OFT} = T_b(log_2 n + 1) + (log_2 n + 1) \times E + 1 \times D + (2log_2 n + 2) \times H$

Broadcast messages in IEEE 802.11 MAC do not use RTS, CTS or ACK. To increase the reliability of broadcast messages,

we can broadcast a message multiple times. In our implementation of the LKH and OFT algorithms, we assume that a message will be broadcast three times for enhanced reliability. Following are the re-keying latencies of the LKH and OFT algorithm in this implementation. (The 802.11 algorithm does not need broadcast messages.)

$J_{LKH-3} = T_u(log_2 n) + 3 \times T_b(log_2 n) + (log_2 n + 1) \times E + log_2 n \times D$

$L_{LKH-3} = 3 \times T_b(2log_2 n) + 2log_2 n \times E + log_2 n \times D$

$J_{OFT-3} = T_u(log_2 n) + 3 \times T_b(log_2 n + 1) + (log_2 n + 1) \times E + log_2 n \times D + (3log_2 n + 2) \times H$

$L_{OFT-3} = 3 \times T_b(log_2 n + 1) + (log_2 n + 1) \times E + 1 \times D + (2log_2 n + 2) \times H$

5) Performance Graphs: To visualize the performance comparison of the three algorithms, we plotted graphs of the above re-keying latency functions $J_{802.11}$, J_{LKH}, J_{LKH-3}, J_{OFT}, J_{OFT_3}, $L_{802.11}$, L_{LKH}, L_{LKH-3}, L_{OFT}, and L_{OFT_3}.

The numerical data for plotting the graphs are given in Table V. We assume AES encryption algorithm [22] with 128-bit keys. The wireless transmission delay components such as DIFS, SIFS, RTS, CTS and ACK are given by the standards [14]– [16].

TABLE V
LATENCY COMPONENTS

Parameter	Description/value	
Spread spectrum technology	OFDM-54 [14]– [16]	DSSS-1 [14]– [16]
T_{SIFS}	9 μs [21]	10 μs [21]
T_{DIFS}	34 μs	50 μs
T_{BO}	67.5 μs	310 μs
T_{RTS}	24 μs	352 μs
T_{CTS}	24 μs	304 μs
T_{ACK}	24 μs	304 μs
$T_{Data}(K)$	$20 + 4 \times \lceil \frac{163+72K}{108} \rceil$	$496 + 144K$
K	Number of keys	
E	Encryption, $2.1ms$ [17], [18]	
D	Decryption, $2.2ms$ [17], [18]	
H	Hashing, $0.009ms$ [19]	

The transmission time $T_{Data}(K)$ of an MSDU depends on its size (or the number of keys it stores) and the data rate at the physical layer (which is determined by the spread spectrum technology), and is calculated as follows [15].

For OFDM-54,

$$T_{Data}(K) = T_{PREAMBLE} + T_{Signal} + T_{SYM} \times \lceil \frac{16 + 6 + 8 \times (34 + MSDU(K))}{N_{DBPS}} \rceil \quad (1)$$

where the service field of the physical layer header is 16 bits long; the PSDU tail of the physical layer header is 6 bits long; the maximum MAC header length is 34 bytes, and $MSDU(K)$ is given by Eq. (1). The values of $T_{PREAMBLE}$, T_{Signal}, T_{SYM} and N_{DBPS} are provided in Table VI.

After substituting the above values into Eq. (2), we obtain:

$$T_{Data}(K) = 20 + 4 \times \lceil \frac{163 + 72K}{108} \rceil \quad (2)$$

as shown in Table V.

TABLE VI
TIMING RELATED PARAMETERS

Parameter	Description	OFDM-54	DSSS-1
$T_{PREAMBLE}$	PLCP preamble duration	16 μs	144 μs
T_{SIGNAL}	Duration of the SIGNAL BPSK-OFDM symbol	4 μs	48 μs
T_{SYM}	Symbol interval	4 μs	
N_{DBPS}	Data bits per OFDM symbol	216 μs	
R_{Data}	Data Rate	$54Mbit/s$	$1Mbit/s$

For DSSS-1,

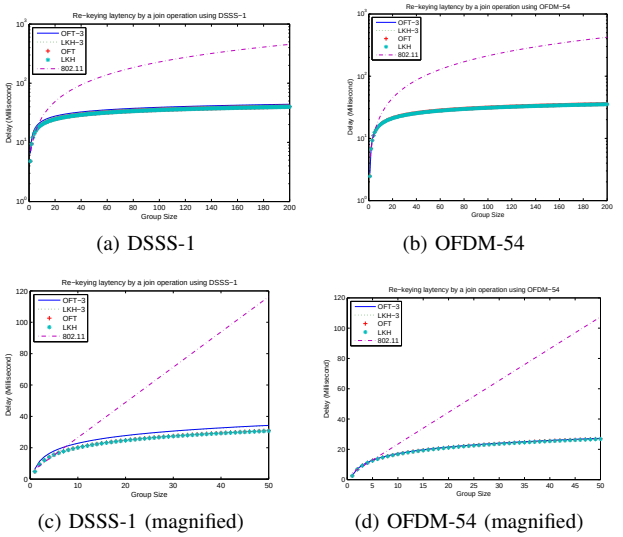
$$T_{Data}(K) = T_{PREAMBLE} + T_{Signal} + \frac{8 \times (34 + MSDU(K))}{R_{Data}} \quad (3)$$

After substituting the appropriate values given in Table VI into the above equation, we obtain:

$$T_{Data}(K) = 496 + 144K \quad (4)$$

After substituting the appropriate values into the latency functions, we plotted the graphs as functions of n, the number of clients in the multicast/broadcast group controlled by an AP. Fig. 7(a) and (b) illustrate the latency functions of the join operation using DSSS-1 and OFDM-54, respectively, on a log scale. To demonstrate the linear function of 802.11 and logarithmic behaviors of LKH and OFT functions, we magnify the above graphs for n values from 0 to 50. The magnified graphs are shown in Fig. 7(c) and (d).

Similarly, the latency functions of the leave operation using DSSS-1 and OFDM-54 are visualized by the graphs in Fig. 8(a) and (b), respectively. The corresponding magnified graphs are given in Fig. 8(c) and (d).



(a) DSSS-1 (b) OFDM-54

(c) DSSS-1 (magnified) (d) OFDM-54 (magnified)

Fig. 7. Rekeying latency incurred by a JOIN operation

6) Discussion: From the above graphs, we draw the following observations.

– The join re-keying latencies of LKH and OFT are comparable, and much lower than that of the 802.11 algorithm when $n > 5$.

– The leave re-keying latency of OFT is slightly better than that of LKH. The reason is that the OFT algorithm incurs 50% less bits to be broadcast for the key update caused by a leave operation [6]. Both algorithms, given their logarithmic running time, perform better than the linear function of 802.11 when $n > 7$.

– Although a message is broadcast three times in the J_{LKH-3}, J_{OFT-3}, L_{LKH-3} and L_{OFT-3} functions for enhanced reliability, the increase of the re-keying latency of this implementation is very minor. This suggests that this implementation is efficient and cost-effective.

Finally, it is worth noting that although the OFT algorithm performs slightly better than the LKH algorithm for leave operations, it is vulnerable to collusion attack [20].

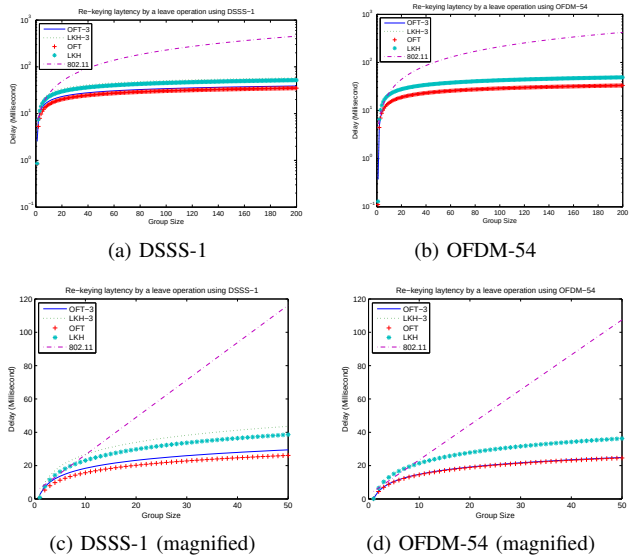
VI. Conclusion

In this paper, we show how the LKH and OFT algorithms can be applied to GKM at the MAC layer in WLANs in order to improve its performance. Our numerical analyses confirm that the LKH and OFT algorithms reduce the re-keying latency of GKM in WLANs from linear time to logarithmic time. Our future work is to evaluate and compare the performance of the 802.11, LKH and OFT algorithms under realistic network settings using simulations and testbeds.

References

[1] C.Wong, M. Gouda, and S. Lam, "Secure Group Communications Using Key Graphs," *IEEE/ACM Transactions Networking*, vol. 8, pp. 16-30, 2000.

[2] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," *Advances in Cryptology*, 1994.



(a) DSSS-1 (b) OFDM-54

(c) DSSS-1 (magnified) (d) OFDM-54 (magnified)

Fig. 8. Rekeying latency incurred by a LEAVE operation

[3] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch Rekeying for Secure Group Communications," *Tenth International World Wide Web Conference*, 2001.

[4] Y. Mao, Y. Sun, M. Wu, and K. J. R. Liu, "Dynamic Join-Exit Amortization and Scheduling for Time-Efficient Group Key Agreement," *IEEE INFOCOMM*, 2004.

[5] D. Balenson, D. McGrew, and A. Sherman, "Key Management for Large Dynamic Groups: One-way Function Trees and Amortized Initialization," IETF Internet Draft: draft-balensongroupkeymgmt-oft-00.txt, 1999.

[6] A. T. Sherman and D. A. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," *IEEE Transactions on Software Engineering*, vol. 29, no. 5, pp. 444-458, 2003.

[7] X. B. Zhang, S. S. Lam, D.-Y. Lee, and Y. R. Yang, "Protocol Design for Scalable and Reliable Group Rekeying," *SPIE Conference on Scalability and Traffic Control in IP Networks*, 2001.

[8] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. "The VersaKey Framework : Versatile Group Key Management,"*IEEE Journal on Selected Areas in Communications (Special Issues on Middleware)*, vol. 17, no. 8, pp. 1614-1631, 1999.

[9] S. Setia, S. Koussih, and S. Jajodia, "Kronos: A Scalable Group Rekeying Approach for Secure Multicast," *IEEE Symposium on Security and Privacy*, 2000.

[10] B. Decleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure Group Communications for Wireless Networks," *IEEE MILCOM*, 2001.

[11] Y. Kim, A. Perrig, and G. Tsudik, "Communication-Efficient Group Key Agreement," *International Information Security Conference*, 2001.

[12] Y. Kim, A. Perrig, and G. Tsudik, "Tree-Based Group Key Agreement," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 60-96, 2004.

[13] D. Steer, L. Strawczynski, W. Diffie, and M. Wiener, "A Secure Audio Teleconference System," *Advances in Cryptology*, 1990.

[14] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Standard 802.11, June 1999.

[15] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification: High-speed Physical Layer Extension in the 2.4 GHz Band, IEEE Standard 802.11, September 1999.

[16] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification: High-speed Physical Layer in the 5Gz Band, IEEE Standard 802.11, September 1999.

[17] http://en.wikipedia.org/wiki/File:Cbc128192256.jpg

[18] A. Sterbenz and P. Lipp. "Performance of the AES Candidate Algorithms in Java," *The Third Advanced Encryption Standard Candidate Conference*, New York, USA, pp. 161-165, 2000.

[19] M. Long, "Energy-efficient and Intrusion Resilient Authentication for Ubiquitous Access to Factory Floor Information," *IEEE Trans. on Industrial Informatics*, pp. 40-47, 2006.

[20] G. Horng, "Cryptanalysis of a Key Management Scheme for Secure Multicast Communications," *IEICE Transactions on Communications*, vol. E85-B, no. 5, pp. 1050-1051, 2002.

[21] Jangeun Jun, Pushkin Peddabachagari and Mihail Sichitiu,"Theoretical Maximum Througput of IEEE 802.11 and its Application," *The 2nd IEEE International Symposium on Network Computing and Applications (NCA'03)*, Cambridge, MA, pp. 249-256, 2003.

[22] G. Berton, "Efficient Software Implementation of AES on 32-bit platforms," *4th International Workshop on Cryptographic Hardware and Embeded System*, 2002.