SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2017, Vol. 60 052109:1–052109:12
doi: 10.1007/s11432-014-0911-8

An efficient protocol for secure multicast key distribution in the presence of adaptive adversaries

Huafei ZHU

School of Computer and Computing Science, Zhejiang University City College, Hangzhou 310015*, China*

Received October 5, 2015; accepted December 4, 2015; published online September 28, 2016

Abstract In this paper, an efficient construction of multicast key distribution schemes based on semantically secure symmetric-key encryption schemes and cryptographically strong pseudo-random number generators is presented and analyzed. The proposed scheme is provably secure against adaptive adversaries leveraging the security amplification technique defined over the logical key hierarchy structures. Our protocol tolerates any coalition of revoked users; in particular, we do not assume any limit on the size or structure of the coalition. The proposed scheme is efficient as a performance of Join or Leave procedure requires $2\log(N)$ multicast activities defined over a sibling ancestor node set, $2\log(N)$ internal state updates of the underlying pseudo-random number generator and $2\log(N)$ symmetric-key encryption activities for N users in a session.

Keywords adaptive adversary, multicast key distribution, pseudo-random number generator, semantic security, symmetric-key encryption

Citation Zhu H F. An efficient protocol for secure multicast key distribution in the presence of adaptive adversaries. Sci China Inf Sci, 2017, 60(5): 052109, doi: 10.1007/s11432-014-0911-8

1 Introduction

The research on group oriented key management has a long history due to its fundamental importance [1]. Existing group key management schemes have been studied mainly within the following two categories referred to as group key agreement schemes and group key distribution schemes:

• A group key agreement scheme allows a group of users to negotiate a common secret key via public channels. Then any group member can encrypt messages with the shared secret key and only the group members can decrypt the generated ciphertexts [2–5].

• A group key distribution scheme allows a trusted and centralized key server called group manager to generate a random key and then allocate it to a group of users. Two different models have been proposed in the literature. One known as broadcast encryption [6–9] assumes that all receivers are stateless and it allows data to be simultaneously delivered to all nodes in a network at a much smaller cost than individually transmitted to each intended recipient. The other usually referred to as multicast key distribution (or group key distribution, or multicast encryption) [10, 11] assumes that receivers maintain their states and it enables a node in a network to efficiently send a single unit of data to an arbitrary set of recipient nodes.

Email: zhuhf@zucc.edu.cn

1.1 Motivation of the problem

It was already mentioned in the paper [12] that a broadcast encryption can be used to establish a group key and thus the state-of-the-art broadcast encryption schemes [7–9] provide feasible solutions to the group key distribution schemes. As a result, one can rely on the GW protocol [8] that tolerates adaptive adversaries with constant size ciphertext and the PPSS protocol [9] that is an extension of BGW [7] to get solutions for group key distribution schemes that tolerate adaptive adversaries. Unfortunately, a rekeying procedure within this framework involves a computation of public keys whose size is linear with the number of group users. As a result, a user must maintain and manage the corresponding private keys whose size is linear with the number of group users. Since the derived group key distribution schemes involve a large number of key sizes, any effort that enables us to tradeoff between the key size (the public and secret key size) and the ciphertext size is certainly welcome.

Alternative solutions to the group key distribution schemes leveraging the logical key hierarchy (LKH) structures have been proposed and analyzed in [10, 11, 13–22] . While extensive researches address the problem of group key distributions, group communication structures, group authentications, revocations and updates of long term keys in the presence of static adversaries [23–27], relatively little work has been reported addressing the constructions of group key distribution protocols tolerating adaptive adversaries within the LKH framework.

Canetti et al. [28] have studied the group key distribution schemes in the presence of adaptive adversaries and they proposed several efficient implementations leveraging the notion of key chains. Unfortunately, Micciancio and Panjwani [29] have shown that the multicast key distribution protocols presented in [28] (and several others, say [10,16]) are insecure in the computational setting (notice that the work of [28] remains secure in the classical Dolev-Yao Model). They further fixed the weakness of these protocols leveraging fully pseudo-random chains (FPC) of keys, a notion similar to that of forward-secure pseudo-random generators [30]. To transmit a key K_t secretly to a user set S_t at time t using the concept of FPC, the group manager first divides the sequence $(1, \ldots, N)$ into the smallest possible of intervals such that every user U_i is contained exactly in an interval and no $U_i \in \overline{S_t}$ is contained in any of intervals, where N denotes the number of group users, $[N] = \{1, \ldots, N\}$, S_t denotes the target set (the targeted receivers) at time t and $\overline{S_t} = [N] \setminus S_t$. Let $I_1, \ldots, I_r$ be these intervals, and for each interval $I_j = (j_1, \ldots, j_m)$ the group manager defines a ciphertext $E_{\overline{K}'_{N-j_1+1}}(E_{K'_{j_m}}(K_t))$, where $(K'_{j_m}, \overline{K}'_{N-j_1+1})$ is derived from a forward chain and a backward chain respectively and is only known by the receivers in I_j. For example, if $N = \{1, \ldots, 11\}$ and $S_t = \{1, 3, 5, 7, 9, 11\}$ at time t, then the number of the intervals is $I_1 = \{1\}$, $I_2 = \{3\}$, $I_3 = \{5\}$, $I_4 = \{7\}$, $I_5 = \{9\}$ and $I_6 = \{11\}$. This means the number of intervals can be the value $|S_t|$. As a result, a rekeying procedure in their remedy scheme requires $O(|S_t|)$ computation and communication complexities. Although, their solution is interesting, the introduced structure of the fully pseudo-random chains of keys (an user interval based solution) is inconsistent with the underlying LKH structure (a key hierarchy based solution) on which the multicast protocols relay. As a result, any solution that is consistent with the LKH structure is certainly welcome.

Xu et al. [19, 20] presented a compiler technique that transforms a tree based group communication scheme secure against static adversaries into the corresponding tree based protocol secure against adaptive adversaries leveraging the notion of pseudo-random functions. Unfortunately, a rekeying procedure in their model requires a refreshment of total internal states of nodes in the current tree. As a result, their scheme requires exact N (N is the number of current group users) computation and communication complexities which could be highly inefficient.

Since the logical key hierarchy structure [10, 13] is extensively used to construct efficient multicast protocols, an interesting research problem thus is how to construct an efficient compiler within the LKH framework that transforms a multicast key distribution protocol tolerating static adversaries to a protocol tolerating adaptive adversaries with $O(\log(N))$ computation and $O(\log(N))$ communication complexities for each rekeying procedure. Equivalently, this paper aims to provide an efficient solution to the following research problem — how to construct efficient multicast key distribution protocols tolerating adaptive adversaries relaying on the logical key hierarchy structure with $O(\log(N))$ computation and $O(\log(N))$

communication complexities for each rekeying procedure?

1.2 This work

In this paper, an efficient construction of multicast key distribution protocols that has logarithmic communication and computation performances in the number of users for Join and Leave operations is proposed and analyzed. In a high level, our protocol works as follows. A group manager creates a logical key hierarchy, where the leaf nodes are associated with the users. Each node is assigned with a random key but only the leaf node keys are given to the corresponding users. Also, each edge is labelled with a value computed by encrypting the parent key as plaintext with the children key. Then with the corresponding leaf key, each user can recursively compute the ancestor keys, including the root key which is used as the group key.

An adversary in our model can corrupt a group user at any session, i.e., an invocation to the Corruption procedure can be launched adaptively. The security of a group key distribution protocol guarantees the forward-security (a revoked user learns nothing about a group key that is generated after an invocation to the Revoke processing) and the backward-security (a newly joined user learns nothing about the group keys generated before she/he joins in the group). To achieve the adaptive security, we first decompose the current key tree into a set of $\log(N)$ subtrees and a path called ancestor node set. We then update the internal states of all roots of the generated subtrees and the internal state of individual node in the ancestor node set. We will get into a bit more detail about the above idea below.

At the initial phase, a group manager (GM) associates individual group user U_i with a leaf node $V_{b_1b_2\cdots b_n}$, where $b_1b_2\cdots b_n$ is an n-bit binary representation of $i \in \mathbb{N}$ which is denoted by $[i]_2 = b_1b_2\cdots b_n$ (we stress that both $V_{b_1b_2\cdots b_n}$ and $U_{b_1b_2\cdots b_n}$ refer to the same user U_i; we use the notation U_i or $U_{b_1b_2\cdots b_n}$ for emphasizing the user itself while we use the notation V_i or $V_{b_1b_2\cdots b_n}$ for emphasizing the location of the user in the tree). By $V_{b_1\cdots b_n} \to V_{b_1b_2\cdots b_{n-1}} \to \cdots \to V_b \to R$, we denote the unique path starting from the leaf node $V_{b_1\cdots b_n}$ to the root R. Let $A_{b_1b_2\cdots b_n} = \{V_{b_1b_2\cdots b_{n-1}}, \ldots, V_{b_1}, R\}$, the set all ancestor nodes of $V_{b_1\cdots b_n}$ and $SA_{b_1b_2\cdots b_n} = \{V_{b_1b_2\cdots\overline{b_{n-1}}}, V_{b_1b_2\cdots\overline{b_{n-2}}}, \ldots, V_{b_1\overline{b_2}}, V_{\overline{b_1}}\}$, the set of sibling ancestor nodes of $V_{b_1b_2\cdots b_n}$. Let $A^+_{b_1b_2\cdots b_n} = A_{b_1b_2\cdots b_n} \bigcup \{V_{b_1b_2\cdots b_n}\}$ and $SA^+_{b_1b_2\cdots b_n} = SA_{b_1b_2\cdots b_n} \bigcup \{V_{b_1b_2\cdots b_{\overline{n}}}\}$, where $b_i \in \{0,1\}$ and $\overline{b_i} = 1 \oplus b_i$. Let $s_{U_i} \leftarrow k_{b_1b_2\cdots b_n}$ where $k_{b_1b_2\cdots b_n}$ is an m-bit random string selected by GM which is then assigned to U_i via a private and authenticated channel established between U_i and GM. Each node $V_{b_1b_2\cdots b_j}$ along the path $A_{b_1b_2\cdots b_n}$ will be assigned a random key value $k_{b_1b_2\cdots b_j}$. The value $e_{b_1b_2\cdots b_j}$ assigned to the edge connecting the adjacent nodes $V_{b_1b_2\cdots b_{j-1}}$ and $V_{b_1b_2\cdots b_j}$ is computed by encrypting the parent key as plaintext with the children key, i.e., $e_{b_1b_2\cdots b_j} = \mathcal{E}_{k_{b_1b_2\cdots b_j}}(k_{b_1b_2\cdots b_{j-1}})$ recursively, where $\mathcal{E}$ refers to a semantically secure symmetric-key encryption scheme (in practice, we assume that the advanced encryption standard [31] achieves the assumed semantic security requirement). The public edge value set $\{e_{b_1b_2\ldots b_n}, e_{b_1b_2\cdots b_{n-1}}, \ldots, e_{b_1}\}$ will be outsourced to the public bulletin board (a publicly accessable database system for all users that is managed and maintained by GM). As long as U_i obtains $pk_{U_i} = \{e_{b_1b_2\cdots b_n}, e_{b_1b_2\cdots b_{n-1}}, \ldots, e_{b_1}\}$ and $sk_{U_i} = k_{b_1b_2\cdots b_n}$, it can compute the ancestor key value $k_{b_1b_2\cdots b_{j-1}} = \mathcal{E}_{k_{b_1b_2\ldots b_j}}(e_{b_1b_2\cdots b_j})$ recursively.

For each update due to a Join or a Leave request, GM performs the following procedures: for a Join request, GM first creates a new key node within the current key tree, and then associates the new user to this created key node; for a Leave request, GM first positions the node location $V_{b_1b_2\cdots b_n}$ and then disassociates this key node with the leaving user. To update the internal states of the related key nodes after a Join association or a Leave disassociation, GM decomposes the key tree into a union of non-overlapped subtrees rooted on $SA^+_{b_1b_2\cdots b_n}$ and the ancestor node set $A^+_{b_1b_2\cdots b_n}$. The internal randomness state and key state of each root are updated by invoking a cryptographically strong pseudo-random number generator. The associated edge values are computed by invoking a semantically secure symmetric-key encryption scheme, and then broadcast the publicly known ciphertexts to all legitimate users whose group keys remain the same at the current rekeying session (note that the generated publicly known data are also stored in the public bulletin board so that all valid uses can access the database to retrieve the related strings). As a result, an update of the entire tree is reduced to that of root nodes in $SA^+_{b_1b_2\cdots b_n}$

and that of key nodes in $A^{+}_{b_1 b_2 \cdots b_n}$ which implies that a performance of Join or Leave procedure requires $2\log(N)$ broadcast activities, $2\log(N)$ internal state updates of the underlying pseudo-random number generator and $2\log(N)$ symmetric-key encryption activities for N users in a session.

1.3 The provable security

We first provide a game-based security definition (for defining the backward and forward security) with adaptive corruptions in the public bulletin board model. We then propose a construction of multicast key distribution schemes based on symmetric-key encryption schemes and pseudo-random number generators. We are able to show that the proposed multicast key distribution scheme tolerating adaptive adversaries if the underlying pseudo-random number generator scheme is cryptographically strong and the symmetric-key encryption scheme is semantically secure. Our method for proving the security leverages the security amplification technique defined over the key distribution tree (The security amplification technique benefits us starting with a weakly secure variant of some cryptographic primitive to build a strongly secure variant of the same primitive. The technique has been extensively used in the cryptography, such as collision-resistant hash functions [32], weakly verifiable puzzles [33–35] and encryption schemes [36]). That is, if an adversary can guess the correct value defined in the security game with a non-negligible advantage then in case that the underlying symmetric-key encryption scheme is semantically secure, either the intermediate key value k_0 of the node V_0 or the key value k_1 of the node V_1 is not uniformly distributed. Recursively, we know that there exists (at least) an intermediate node such that the key value of that intermediate node is not uniformly distributed. Since all intermediate nodes are managed and maintained by the group manager that cannot be corrupted by assumption, we know that there exists at least one leaf node whose key value is not uniformly distributed from the point view of the adversary. If such a leaf node can be positioned with a non-negligible advantage (we will use the security amplification technique to position such a leaf node set, see the proof of Lemma 1 for more details), then we are able to construct a probabilistic polynomial time (PPT) distinguisher for the underlying pseudo-random number generator, from which we arrive at a contradiction.

1.4 Summarization

In summary, the contribution of this paper comprises the following three-fold:

• In the first fold, a game-based security definition (for defining the backward and forward security) relaying on the group key decomposition technique with adaptive corruptions in the logical key hierarchy structure is introduced and formalized.

• In the second fold, an efficient construction of multicast key distribution protocols is proposed based on the key tree decomposition technique introduced in this paper with $2\log(N)$ computation and $2\log(N)$ communication complexities for each rekeying procedure.

• In the third fold, a rigorous security proof leveraging the security amplification technique defined over the logical key hierarchy structure is presented. We show that the proposed multicast key distribution protocol tolerates adaptive adversaries assuming that the underlying pseudo-random number generator is cryptographically strong and the symmetric-key encryption scheme is semantically secure.

The novelty of this paper comes from the following facts:

• The novelty of protocol construction. A cryptographically secure pseudo-random number generator (PRNG) is defined by the group manager and is shared among all leaf nodes and internal nodes for updating their internal states (randomness states and key states). A description of the specified PRNG is then posted on the public bulletin board (a similar idea with the common reference string model extensively used in the literature). Since the guarantee of a PRNG is that a single output appears random if the input was chosen at random, the application of PRNG is sufficient for each internal states update which is more convenient compared with the known results where individual leaf node is equipped with a pseudo-random function (PRF). Since the guarantee of a PRF is that all its outputs appear random as long as the function was drawn at random from the PRF family, it follows that the previously known PRF based solutions is more complicated.

• The novelty of key state update. Another novel technique introduced in this paper is the entire key tree decomposition methodology. We decompose the entire tree into a union of non-overlapped subtrees rooted on $SA^{+}_{b_1b_2\ldots b_n}$ and the ancestor node set $A^{+}_{b_1b_2\ldots b_n}$ for any node $V_{b_1b_2\ldots b_n}$ and then update in the corresponding nodes in $SA^{+}_{b_1b_2\ldots b_n}$ and in $A^{+}_{b_1b_2\ldots b_n}$. Since the non-overlapped subtrees support the parallel computations defined over $SA^{+}_{b_1b_2\ldots b_n}$ and in $A^{+}_{b_1b_2\ldots b_n}$, it follows that our protocol is suitable for a large number of group users.

• The novelty of security proof. As far as we can tell, the security amplification technique is first applied to the security proof of the multicast key distributions. Our security proof technique demonstrates the power of the security amplification technique. This could be interesting from the point of view of the theoretical research.

The rest of this paper is organized as follows: in Section 2, syntax and security definitions of multicast key distribution protocols in the presence of adaptive adversaries are presented. A construction of multicast key distribution protocols leveraging the notions of semantically secure symmetric-key encryption schemes and cryptographically strong pseudo-random number generators is then proposed and analyzed in Section 3. We conclude this work in Section 4.

2 Multicast key distribution: syntax and security definition

In this section, syntax and security definition of multicast key distribution protocols are introduced and formalized in the standard computation complexity model, where all participants are modelled as interactive Turing machines. We assume that users' public data (user id, edge values and other related public data) are stored in a public bulletin board and all messages posted to the bulletin board are authenticated. Usually, it is assumed that any data written to the bulletin board cannot be erased or tampered with. The syntax and security definition of multicast key distributions below are mainly due to [12, 20, 28, 29] but they are tailored for the key tree decomposition technique introduced in this paper.

2.1 Syntax of multicast key distributions

Let PBB be a public bulletin board available for all users. A multicast key distribution protocol which is managed, maintained by a primary user called group manager (GM), consists of the following four algorithms:

• Setup. This algorithm is executed by GM at the initial session sid = 0. GM takes as input a security parameter 1^{κ}, an initial user set $U^{(0)}$ and outputs a group of keys $K^{(0)}_{\mathrm{GM}} = (pk^{(0)}_{\mathrm{GM}}, sk^{(0)}_{\mathrm{GM}})$ for $U^{(0)}$, where $pk^{(0)}_{\mathrm{GM}} = \bigcup_{i=1}^{|U^{(0)}|} pk^{(0)}_{U}$, $sk^{(0)}_{\mathrm{GM}} = \bigcup_{i=1}^{|U^{(0)}|} sk^{(0)}_{U}$, $pk^{(0)}_{U}$ defines user's public string and $sk^{(0)}_{U}$ denotes the corresponding secret key in the initial session sid = 0. The public information $pk^{(0)}_{\mathrm{GM}}$ is stored at PBB. GM keeps the initial private keys $sk^{(0)}_{\mathrm{GM}}$ secret and stores them at her private database.

• Join. This algorithm is executed by GM at session sid = l due to a Join request. GM takes as input identities of the previous group members $U^{(l-1)}$, identity of a new member $U' \notin U^{(l-1)}$, $K^{(l-1)}_{\mathrm{GM}}$ and outputs $(U^{(l)}, K^{(l)}_{\mathrm{GM}})$. The public information $pk^{(l)}_{\mathrm{GM}}$ is stored at PBB. GM keeps the updated secret keys $sk^{(l)}_{\mathrm{GM}}$ secret and stores them at her private database.

• Leave. This algorithm is executed by GM at session sid = l due to a Leave request. GM takes as input the identities of the previous group members $U^{(l-1)}$, identity of a leaving member $U' \in U^{(l-1)}$, $K^{(l-1)}_{\mathrm{GM}}$ and outputs $(U^{(l)}, K^{(l)}_{\mathrm{GM}})$. The public information $pk^{(l)}_{\mathrm{GM}}$ is stored at PBB. GM keeps the updated secret keys $sk^{(l)}_{\mathrm{GM}}$ secret and stores them at her private database.

• Rekeying. This algorithm is executed by a legitimate group member U at session sid = l which takes as input $(pk^{(l)}_{U}, sk^{(l)}_{U}) \in K^{(l)}_{\mathrm{GM}}$ and outputs the current common group key $k^{(l)} \in \bigcap_{i=1}^{|U^{(l)}|} sk^{(l)}_{U}$.

Definition 1. The correctness of a multicast key distribution protocol means that for an arbitrary session sid = l and an arbitrarily legitimate user $U \in U^{(l)}$ at the current session, GM and U share the same group key $k^{(l)}$ if the user U is honest.

2.2 The definition of security

To define the security of the multicast key distribution protocols, we consider an adversary who controls over all communications within the networks. The adversary in our model is allowed to corrupt a user at any session and to learn all internal states once a user is corrupted. The adversary's interaction with parties is modelled by allowing it to have access to the following oracles:

• $\mathcal{O}^{\text{Setup}}$. On input (Setup, sid = 0, $U^{(0)}$), $\mathcal{O}^{\text{Setup}}$ outputs ($pk_{\text{GM}}^{(0)}$, $sk_{\text{GM}}^{(0)}$). The adversary is given $pk_{\text{GM}}^{(0)}$.

• $\mathcal{O}^{\text{Corrupt}}$. Let $H^{(l-1)}$ be a set of honest users and $C^{(l-1)}$ be a set of corrupted users at session sid $= l-1$. On input (Corrupt, sid $= l$, $U^{(l-1)}$, $U' \in H^{(l-1)}$), $\mathcal{O}^{\text{Corrupt}}$ outputs ($sk_{U'}^{(l-1)}$, $pk_{\text{GM}}^{(l-1)}$), $H^{(l)} = H^{(l-1)} \setminus \{U'\}$ and $C^{(l)} = C^{(l-1)} \bigcup \{U'\}$. The adversary is given ($sk_{U'}^{(l-1)}$, $pk_{\text{GM}}^{(l-1)}$), $H^{(l)}$ and $C^{(l)}$.

• $\mathcal{O}^{\text{Join}}$. On input (Join, sid $= l$, $U^{(l-1)}$, $U' \notin U^{(l-1)}$, $K_{\text{GM}}^{(l-1)}$), $\mathcal{O}^{\text{Join}}$ outputs $U^{(l)}$ and ($pk_{\text{GM}}^{(l)}$, $sk_{U'}^{(l)}$). The adversary is given $U^{(l)}$ and $pk_{\text{GM}}^{(l)}$.

• $\mathcal{O}^{\text{Leave}}$. On input (Leave, sid $= l$, $U^{(l-1)}$, $U' \in U^{(l-1)} \bigcap H^{(l-1)}$, $K_{\text{GM}}^{(l-1)}$), $\mathcal{O}^{\text{Leave}}$ outputs ($U^{(l)}$, $K_{\text{GM}}^{(l)}$). The adversary is given $U^{(l)}$ and $pk_{\text{GM}}^{(l)}$.

• $\mathcal{O}^{\text{Revoke}}$. On input (Revoke, sid $= l$, $U^{(l-1)}$, $U' \in U^{(l-1)} \bigcap C^{(l-1)}$, $K_{\text{GM}}^{(l-1)}$), $\mathcal{O}^{\text{Revoke}}$ outputs ($U^{(l)}$, $K_{\text{GM}}^{(l)}$) and $sk_{U'}^{(l-1)}$. The adversary is given $U^{(l)}$, $pk_{\text{GM}}^{(l)}$ and $sk_{U'}^{(l-1)}$.

Definition 2 (Backward-security against an adaptive adversary)**.** Consider the following experiment with total L oracle queries (L serves as an upper bound of the oracle queries):

• The adversary $\mathcal{A}$ initially performs the oracle query $\mathcal{O}^{\text{Setup}}$ and obtains $U^{(0)}$ and $pk_{\text{GM}}^{(0)}$.

• In each session, $\mathcal{A}$ performs one of oracle queries: $\mathcal{O}^{\text{Join}}$, $\mathcal{O}^{\text{Leave}}$, $\mathcal{O}^{\text{Corrupt}}$, or $\mathcal{O}^{\text{Revoke}}$ in an arbitrary order.

If $\mathcal{A}$ performs an oracle query $\mathcal{O}^{\text{Revoke}}$ at session sid $= l_1$, then $\mathcal{O}^{\text{Corrupt}}$ should be queried at any session sid $= l_2$ such that $1 \leqslant l_2 < l_1 < L-1$.

If an oracle query $\mathcal{O}^{\text{Corrupt}}$ has been processed at session l_1, then an oracle query $\mathcal{O}^{\text{Revoke}}$ should be performed at any later session sid $= l_2$ such that $l_1 < l_2 < L-1$.

• Let session id sid $= l_3$ be the last session performing $\mathcal{O}^{\text{Revoke}}$ query. This captures the requirement that all corrupted members should be revoked before a challenging group key (at the session sid $= L$) is given.

At each session sid $= t$ ($l_3 + 1 \leqslant t \leqslant L-1$), $\mathcal{A}$ performs an arbitrarily oracle query either $\mathcal{O}^{\text{Join}}$ or $\mathcal{O}^{\text{Leave}}$ but neither $\mathcal{O}^{\text{Corrupt}}$ or $\mathcal{O}^{\text{Revoke}}$ is invoked. The output of each oracle query is $U^{(t)}$ and ($pk_{\text{GM}}^{(t)}$, $sk_{\text{GM}}^{(t)}$). The adversary is given $U^{(t)}$ and $pk_{\text{GM}}^{(t)}$.

• Finally, $\mathcal{A}$ queries the oracle $\mathcal{O}^{\text{Join}}$ at session sid $= L$ and $\mathcal{A}$ is given $U^{(L)}$ and ($pk_{\text{GM}}^{(L)}$, $sk_{\mathcal{A}}^{(L)}$).

• $\mathcal{A}$ now is given two m-bit strings $k_0^{(t)}, k_1^{(t)}$, where $k_b^{(t)}$ is a group key at session sid $= t$, $k_{\overline{b}}^{(t)}$ is a random m-bit string and $l_3 + 1 \leqslant t \leqslant L-1$. Finally, $\mathcal{A}$ outputs a bit b'.

Let $\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[b = b'] - 1/2|$. A multicast key distribution protocol achieves the backward-security in the presence of an adaptive adversary $\mathcal{A}$ if $\text{Adv}_{\mathcal{A}}(\kappa)$ is at most a negligible amount.

Definition 3 (Forward-security against an adaptive adversary)**.** Consider the following experiment with total L oracle queries:

• The adversary $\mathcal{A}$ initially performs the oracle query $\mathcal{O}^{\text{Setup}}$ and obtains $U^{(0)}$ and $pk_{\text{GM}}^{(0)}$.

• At each session, $\mathcal{A}$ performs one of the oracle queries: either $\mathcal{O}^{\text{Join}}$, or $\mathcal{O}^{\text{Leave}}$, or $\mathcal{O}^{\text{Corrupt}}$, or $\mathcal{O}^{\text{Revoke}}$ in an arbitrary order.

If $\mathcal{A}$ performs an oracle query $\mathcal{O}^{\text{Revoke}}$ at session sid $= l_1$, then $\mathcal{O}^{\text{Corrupt}}$ should be performed at a session sid $= l_2$ such that $1 \leqslant l_2 < l_1 < L-1$.

If $\mathcal{A}$ performs an oracle query $\mathcal{O}^{\text{Corrupt}}$ at session l_1, then $\mathcal{O}^{\text{Revoke}}$ should be performed at a session sid $= l_2$ such that $1 \leqslant l_1 < l_2 < L-1$.

• Let sid $= l_3$ be the last session performing the $\mathcal{O}^{\text{Revoke}}$ query. This captures the requirement that all corrupted members should be revoked before a challenging group key is given.

At each session sid $= t$ where $l_3 + 1 \leqslant t \leqslant L - 1$, $\mathcal{A}$ performs one of oracle queries: either $\mathcal{O}^{\text{Join}}$ or $\mathcal{O}^{\text{Leave}}$ but neither $\mathcal{O}^{\text{Corrupt}}$ nor $\mathcal{O}^{\text{Revoke}}$ in an arbitrary order.

• Finally, $\mathcal{A}$ queries $\mathcal{O}^{\text{Leave}}$ at session sid $= L$. $\mathcal{A}$ is given $U^{(L)}$ and $pk_{\text{GM}}^{(L)}$; $\mathcal{A}$ now is given two m-bit strings $k_0^{(L)}, k_1^{(L)}$, where $k_b^{(L)}$ is a group key at session sid $= L$ while $k_{\overline{b}}^{(L)}$ is a random m-bit string. Finally, $\mathcal{A}$ outputs a bit b'.

Let $\text{Adv}_{\mathcal{A}}(\kappa) = |\Pr[b = b'] - 1/2|$. A multicast key distribution protocol achieves the forward-security in the presence of an adaptive adversary $\mathcal{A}$ if $\text{Adv}_{\mathcal{A}}(\kappa)$ is at most a negligible amount.

Definition 4. A multicast key distribution protocol is secure against adaptive adversaries if it achieves the forward and backward security in the presence of adaptive adversaries.

3 Multicast system: construction and proof of security

Our multicast key distribution protocol is constructed from the logical key hierarchy structure based on semantically secure symmetric-key encryptions and cryptographically strong pseudo-random number generators. We assume that the reader is familiar with the standard security definitions of symmetric-key encryptions [37] and pseudo-random number generators [38]. The syntax and security definitions of the underlying cryptographic primitives are thus omitted.

3.1 The construction

Let G be a cryptographically strong pseudo-random number generator which takes as input $s' \in \{0,1\}^m$ and outputs $2m$-bit strings (s, k), where $s, k \in \{0,1\}^m$, s stands for the randomness state and k defines the key state. Let $y = \{x\}_k$ be an encryption of message x under the secret key k using semantically secure symmetric-key encryption scheme (say, AES). Let $U_i^{(j)}$ be the id of user U_i at session sid $= j$ ($U_i^{(j)}$ and U_i are identical at session sid $= j$). A construction of our multicast key distribution protocol comprises the following steps:

• Setup. At the initial session sid $= 0$, the group manager (GM) assigns an initial user $U_i^{(0)}$ to a leaf node $V_{b_1b_2\cdots b_n}$, where $[i]_2 = b_1b_2\cdots b_n$. GM then generates a pair of public edge value and secret key for each internal key node $V_{b_1\cdots b_j}$ $(j = 1, \ldots, n)$ according to the following steps.

- GM chooses an m-bit string $s'^{(0)}_{b_1b_2\cdots b_j}$ uniformly at random on behalf of the key node $V_{b_1\cdots b_j}$ for $j = 1, \ldots, n$.

- GM then invokes G which takes $s'^{(0)}_{b_1b_2\cdots b_j}$ as input to generate randomness state and internal key state, i.e., $(s^{(0)}_{b_1b_2\cdots b_j}, k^{(0)}_{b_1b_2\cdots b_j}) = G(s'^{(0)}_{b_1b_2\cdots b_j})$, where $s^{(0)}_{b_1b_2\cdots b_j}$ is the initial randomness state and $k^{(0)}_{b_1b_2\ldots b_j}$ is the initial key state. GM keeps the seed $s'^{(0)}_{b_1b_2\cdots b_j}$ secret.

Let $A_{b_1b_2\cdots b_n} = \{V_{b_1b_2\cdots b_{n-1}}, \ldots, V_{b_1}, R\}$ be the ancestor set of $V_{b_1\cdots b_n}$, where R is the root of the current tree. Let $A^+_{b_1b_2\cdots b_n} = A_{b_1b_2\cdots b_n} \cup \{V_{b_1b_2\ldots b_n}\}$. The public edge values along $A^+_{b_1b_2\cdots b_n}$ are computed by GM as follows:

- $e^{(0)}_{b_1b_2\cdots b_n} = \{k^{(0)}_{b_1b_2\cdots b_{n-1}}\}_{k^{(0)}_{b_1b_2\cdots b_n}}$;

- $\cdots$;

- $e^{(0)}_{b_1} = \{k^{(0)}\}_{k^{(0)}_{b_1}}$, where $k^{(0)} \in \{0,1\}^m$ is an output of G that takes as input $k'^{(0)} \in_U \{0,1\}^m$ ($k'^{(0)}$ is called an initial seed of G which is selected uniformly at random in $\{0,1\}^m$) to output a randomness and a key state $(s^{(0)}, k^{(0)})$ and the derived key state $k^{(0)}$ is defined as initial group key by GM.

Let $pk^{(0)}_{b_1b_2\cdots b_n} = \{e^{(0)}_{b_1b_2\cdots b_n}, e^{(0)}_{b_1b_2\cdots b_{n-1}}, \ldots, e^{(0)}_{b_1}\}$ be the public information (or string or public edge value) and $sk^{(0)}_{b_1b_2\cdots b_n} = \{(k^{(0)}_{b_1b_2\cdots b_n}, s^{(0)}_{b_1b_2\cdots b_n})\}$ be the secret key of $V_{b_1\ldots b_n}$. The user $U_i^{(0)}$ is then given $(pk^{(0)}_{b_1b_2\cdots b_n}, sk^{(0)}_{b_1b_2\cdots b_n})$.

• Join. This algorithm is executed by the group manager at session sid $= l$ due to a Join request. When GM receives a Join request, it takes as input the identities of the previous group members $U^{(l-1)}$, a new user U' and the state set $(pk^{(l-1)}_{\text{GM}}, sk^{(l-1)}_{\text{GM}})$ and outputs $U^{(l)}$, $(pk^{(l)}_{\text{GM}}, sk^{(l)}_{\text{GM}})$, $l \geqslant 1$. More precisely,

- GM creates a new leaf node $V_{b_1b_2\cdots b_n}$ for U'. The user node U' is then identical with the leaf node $V_{b_1b_2\cdots b_n}$ accordingly (for simplicity, we identify U' with $U^{(l)}_{b_1b_2\cdots b_n}$ which in turn is identical with the leaf node $V_{b_1b_2\cdots b_n}$). GM then assigns a random seed $s'^{(l)}_{b_1b_2\cdots b_n}$ to $U^{(l)}_{b_1b_2\cdots b_n}$. The user node $U^{(l)}_{b_1b_2\cdots b_n}$ then invokes G which takes as input $s'^{(l)}_{b_1b_2\cdots b_n}$ to output $(s^{(l)}_{b_1b_2\cdots b_n}, k^{(l)}_{b_1b_2\cdots b_n})$ for the current session sid $= l$.

- Let $SA_{b_1b_2\cdots b_n} = \{V_{b_1b_2\cdots\overline{b_{n-1}}}, V_{b_1b_2\cdots\overline{b_{n-2}}}, \ldots, V_{b_1\overline{b_2}}, V_{\overline{b_1}}\}$ be the sibling ancestor set of $V_{b_1b_2\cdots b_n}$ and $SA^+_{b_1b_2\cdots b_n}$=$SA_{b_1b_2\cdots b_n} \cup \{V_{b_1b_2\cdots\overline{b_n}}\}$. GM updates the internal states of all nodes in $A^+_{b_1b_2\cdots b_n}$ and $SA^+_{b_1b_2\cdots b_n}$ according to the following procedures:

- For each key node $V_{b_1\ldots b_j} \in A^+_{b_1b_2\cdots b_n}$ $(j = 1, \ldots, n)$, we must consider the following two cases: 1) $V_{b_1\cdots b_j}$ exists before the current session takes place (old node mode) and 2) $V_{b_1\cdots b_j}$ is a newly created internal node (new node mode). We remark that the definition of new node mode is needed for us to consider a scenario where $U^{(l-1)} = (V_{00}, V_{01})$, U'=V_{10}, as such V_1 is a newly created node and the corresponding internal key state $s^{(l-1)}_1$ has not been defined, and thus more clarification is needed here. If $V_{b_1\cdots b_j}$ is an old internal node, GM first invokes G which takes as input $s^{(l-1)}_{b_1\cdots b_j}$ (the randomness state of this internal node at session sid $= l-1$) to output the randomness state and key state $(s^{(l)}_{b_1\cdots b_j}, k^{(l)}_{b_1\cdots b_j}) = G(s^{(l-1)}_{b_1\cdots b_j})$. If $V_{b_1\ldots b_j}$ is a new internal node, GM assigns a random seed $s'^{(l)}_{b_1b_2\cdots b_j}$ which is selected uniformly at random from $\{0,1\}^m$ to the newly created internal node $V^{(l)}_{b_1b_2\cdots b_j}$ and then invokes G which takes as input $s'^{(l)}_{b_1b_2\cdots b_j}$ to output $(s^{(l)}_{b_1b_2\cdots b_j}, k^{(l)}_{b_1b_2\cdots b_j})$ for the current session sid $=l$. The corresponding edge values (for both cases) are computed as follows: $e^{(l)}_{b_1b_2\cdots b_j} = \{k^{(l)}_{b_1b_2\cdots b_{j-1}}\}_{k^{(l)}_{b_1b_2\cdots b_j}}, \ldots, e^{(l)}_{b_1} = \{k^{(l)}\}_{k^{(l)}_{b_1}}$, where GM invokes G which takes $s^{(l-1)}$ as an input to output $(s^{(l)}, k^{(l)})$ and $k^{(l)}$ serves as the group key specified by GM at the current session. The public information of $V_{b_1\cdots b_j}$ is $pk^{(l)}_{b_1b_2\cdots b_j} = \{e^{(l)}_{b_1b_2\cdots b_j}, e^{(l)}_{b_1b_2\cdots b_{j-1}}, \ldots, e^{(l)}_{b_1}\}$ and the secret key is $sk^{(l)}_{b_1b_2\cdots b_j} = \{(k^{(l)}_{b_1b_2\cdots b_j}, s^{(l)}_{b_1b_2\cdots b_j})\}$.

- For each key node $V \in SA^+_{b_1b_2\cdots b_n}$, GM first updates the randomness state and internal state of V by applying the general procedure defined above (for computing the edge value of the key node $V_{b_1\cdots b_j} \in A^+_{b_1b_2\cdots b_n}$ either in the old node mode or the new node mode). GM then defines the unique subtree of the current entire tree rooted on V. Let $T^{(l)}_{\overline{b_1}}$ be the subtree rooted on $V_{\overline{b_1}}$ at the session sid $= l$ and $e^{(l)}_{\overline{b_1}}$ be the value of edge connecting $V_{\overline{b_1}}$ and its parent node R that is computed by $\{k^{(l)}\}_{k^{(l)}_{\overline{b_1}}}$ $(= e^{(l)}_{\overline{b_1}})$. Given $e^{(l)}_{\overline{b_1}}$, GM further computes values $\{e^{(l)}_{\overline{b_10}}, e^{(l)}_{\overline{b_11}}\}$ of edges connecting $V_{\overline{b_1}}$ with its children $V_{\overline{b_10}}$ and $V_{\overline{b_11}}$ by applying the general procedure defined above. Finally, GM broadcasts the encrypted edge values $\{e^{(l)}_{\overline{b_1}}, e^{(l)}_{\overline{b_10}}\}_{k^{(l-1)}_{\overline{b_10}}}$ to all leaf nodes within the subtree $T^{(l)}_{\overline{b_10}}$ and the encrypted edge values $\{e^{(l)}_{\overline{b_1}}, e^{(l)}_{\overline{b_11}}\}_{k^{(l-1)}_{\overline{b_11}}}$ to all leaf nodes within the subtree $T^{(l)}_{\overline{b_11}}$. Since all key nodes within the subtree $T^{(l)}_{\overline{b_10}}$ ($T^{(l)}_{\overline{b_11}}$, resp.) share the common key state $k^{(l-1)}_{\overline{b_10}}$ ($k^{(l-1)}_{\overline{b_11}}$, resp.), all leaf nodes within the corresponding subtree can decrypt the related ciphertexts and thus learn all the modified edge values. That is, all key leaf nodes within this subtree are able to compute the updated values of edges. It follows that the public information $pk^{(l)}_V$ and the secret key $sk^{(l)}_V$ of V are well defined. This procedure continues until all nodes in $SA^+_{b_1b_2\cdots b_n}$ are updated.

- For each node neither in $A^+_{b_1b_2\cdots b_n}$ nor in $SA^+_{b_1b_2\cdots b_n}$, the internal states (the internal randomness state and the key state) remain the same.

• Leave. This algorithm is executed by the group manager at session sid $= l$ due to a Leave request. GM takes as input the identities of the previous group members $U^{(l-1)}$, identity of a leaving member $U^{(l-1)}_{b_1b_2\cdots b_n}$ and public strings and secret keys $(pk^{(l-1)}_{\mathrm{GM}}, sk^{(l-1)}_{\mathrm{GM}})$ of the current key tree to output $(pk^{(l)}_{\mathrm{GM}}, sk^{(l)}_{\mathrm{GM}})$. That is,

- GM virtually disconnects the leaving user node $U^{(l-1)}_{b_1b_2\cdots b_n}$ with the associated node $V_{b_1b_2\cdots b_n}$. Then GM updates the internal states of each node in $A^+_{b_1b_2\cdots b_n}$ and $SA^+_{b_1b_2\cdots b_n}$ and finally updates the edge values as that described in the Join procedure.

- For each node neither in $A^+_{b_1b_2\cdots b_n}$ nor in $SA^+_{b_1b_2\cdots b_n}$, GM updates its session id sid $= l$ while the internal states (the randomness state and the key state) remain the same. At the end of this procedure,

the public information $pk_{\mathrm{GM}}^{(l)}$ and the corresponding secret key $sk_{\mathrm{GM}}^{(l)}$ of the new group are defined.

• Rekeying. This algorithm is executed by the legitimate group members at session sid $= l$ when either a Join request or a Leave request is issued. Rekeying takes as input $pk_{U_{b_1\cdots b_n}}^{(l)}$ and $sk_{U_{b_1\cdots b_n}}^{(l)}$ and outputs the common group key $k^{(l)}$ at session sid $= l$.

One can verify that a performance of Join or Leave procedure requires $2\log(N)$ broadcast activities, $2\log(N)$ internal state updates of the underlying pseudo-random number generator and $2\log(N)$ symmetric-key encryption activities for N users in a session.

3.2 The proof of security

The correctness of the proposed protocol can be verified step by step according to the procedure depicted above and thus the details are omitted. In the rest of this section, we will prove our main result stated in Theorem 1.

Lemma 1. The proposed multicast key distribution protocol achieves the backward-security in the presence of adaptive adversaries assuming that the underlying pseudo-random number generator is cryptographically strong and the symmetric-key encryption scheme is semantically secure.

Proof. Let π be the protocol depicted in Subsection 3.1. Let G_0 be the game for defining the backward security. Suppose that the adversary's advantage defined in G_0 is ϵ. Let $\epsilon = (1-\alpha)^s + \beta$, where $\alpha, \beta \in (0,1)$ and s is the number of leaf nodes defined by the adversary's strategy. By $(V_{i_1}, V_{i_2}, \ldots, V_{i_s}) \propto \mathcal{D}$, we denote an event that the adversary is able to guess the correct value b defined in the game G_0 based on a set of s leaf nodes $(V_{i_1}, V_{i_2}, \ldots, V_{i_s})$ which is selected according to the adversary's strategy. For a leaf node index $i \in [N]$, we say i is $j-$Leak, if when the j-th leaf index is i and all the others are chosen according to the adversary's strategy, the adversary is able to guess the correct value b with high enough probability, where N is the number of group users. That is, i is $j-$Leak if and only if $\Pr_{i_1,\ldots,i_s}[(V_{i_1}, \ldots, V_{i_{j-1}}, V_i, V_{i_{j+1}}, \ldots, V_{i_s}) \propto \mathcal{D}] \geqslant \beta/s$.

Let $\mathrm{Leak}_j = \{i \in [N] \mid i \text{ is } j-\mathrm{Leak}\}$. We want to show that $|\mathrm{Leak}_j|/N$ is at least $1-\alpha$ for some $j \in [s]$. Suppose that for all $j \in [s]$, we have $|\mathrm{Leak}_j|/N < 1-\alpha$. We consider the following probability statement according to the adversary's strategy:

$$
\begin{aligned}
\epsilon &= \Pr_{i_1,\ldots,i_s}[(V_{i_1}, \ldots, V_{i_{j-1}}, V_{i_j}, V_{i_{j+1}} \ldots, V_{i_s}) \propto \mathcal{D}] \\
&= \Pr_{i_1,\ldots,i_s}[(V_{i_1}, \ldots, V_{i_s}) \propto \mathcal{D} \wedge \text{ all } i_m \text{ are } j-\mathrm{Leak}] \\
&\quad + \Pr_{i_1,\ldots,i_s}[(V_{i_1}, \ldots, V_{i_s}) \propto \mathcal{D} \wedge \text{ not all } i_m \text{ are } j-\mathrm{Leak}] \\
&\leqslant \Pr_{i_1,\ldots,i_s}[\text{ all } i_m \text{ are } j-\mathrm{Leak}] \\
&\quad + \Pr_{i_1,\ldots,i_s}[(V_{i_1}, \ldots, V_{i_s}) \propto \mathcal{D} \wedge \text{ exits } i_m \text{ such that } i_m \text{ is not } j-\mathrm{Leak}] \\
&= \Pr_{i_1,\ldots,i_s}[\text{ all } i_m \text{ are } j-\mathrm{Leak}] \\
&\quad + \sum_{m=1}^{s} \Pr_{i_1,\ldots,i_s}[(V_{i_1}, \ldots, V_{i_s}) \propto \mathcal{D} \wedge \ i_m \text{ is not } j-\mathrm{Leak}] \\
&\leqslant \Pr_{i_1,\ldots,i_s}[\text{ all } i_m \text{ are } j-\mathrm{Leak}] \\
&\quad + \sum_{m=1}^{s} \Pr_{i_1,\ldots,i_s}[(V_{i_1}, \ldots, V_{i_s}) \propto \mathcal{D} |\ i_m \, is \text{ not } j-\mathrm{Leak}] \\
&< (1-\alpha)^s + s \times \frac{\beta}{s} = (1-\alpha)^s + \beta.
\end{aligned}
$$

We have a contradiction. For j =1 to s, for i =1 to N, we repeat $s([\ln(N)]+1)/\beta$ times by choosing leaf vector $(i_1, \ldots, i_{j-1}, i_j, i_{j+1}, \ldots, i_s)$ and replacing $(i_1, \ldots, i_{j-1}, i_j, i_{j+1}, \ldots, i_s)$ with $(i_1, \ldots, i_{j-1}, i, i_{j+1}, \ldots, i_s)$ and querying the validity of the event $(V_{i_1}, \ldots, V_{i_s}) \propto \mathcal{D}$. Consequently, if i is $j-$Leak then during the loop with (j, i), the adversary will return a valid answer with probability at least $1-(1-\frac{\beta}{s})^{s([\ln(N)]+1)/\beta}$. As a result, we are able to position a pair of leaf indexes (i, j) such that i is a $j-$Leak with probability $1-\frac{1}{eN}$.

In the following, we will make use of the extracted index pair (i, j) to distinguish the challenging string $k \in \{0,1\}^m$ which is either a random string or an output of the pseudo-random number generator G in the special case where $s = 1$. Our simulation (Game G_1) is described as follows:

• Whenever the adversary $\mathcal{A}$ queries a setup oracle $\mathcal{O}^{\text{Setup}}$, the simulator selects a random string $s'^{(0)} \in \{0,1\}^m$ uniformly at random, invokes G which takes as input $s'^{(0)}$ to output $(k^{(0)}, s^{(0)})$, where $k^{(0)}$ is an initial key state and $s^{(0)}$ is an initial state of the root R. The answer to the oracle query $\mathcal{O}^{\text{Setup}}$ is the same as that described by the Setup procedure specified in the protocol π.

• Whenever the adversary $\mathcal{A}$ queries either $\mathcal{O}^{\text{Join}}$, or $\mathcal{O}^{\text{Leave}}$, or $\mathcal{O}^{\text{Corrupt}}$, or $\mathcal{O}^{\text{Revoke}}$ in an arbitrary order, the answer to each oracle is the same as that described by the corresponding procedure (either Join, or Leave, or Corrupt, or Revoke) specified in the protocol π. We emphasize that if $\mathcal{A}$ performs an oracle query $\mathcal{O}^{\text{Revoke}}$ at session sid $= l_1$ on behalf of a corrupted user U', then $\mathcal{O}^{\text{Corrupt}}$ must be performed at some session sid $= l_2$ on behalf of the same U', where $1 \leqslant l_2 < l_1 < L - 1$. We also emphasize that if an oracle query $\mathcal{O}^{\text{Corrupt}}$ on U' has been processed at session l_1, then an oracle query $\mathcal{O}^{\text{Revoke}}$ should be performed at session sid $= l_2$, where $l_1 < l_2 < L - 1$.

Let session id sid $= l_3$ be the last session processing $\mathcal{O}^{\text{Revoke}}$. The simulator now randomly selects a session sid $= t$ $(l_3 + 1 \leqslant t \leqslant L - 1)$ and then performs arbitrarily an oracle query $\mathcal{O}^{\text{Join}}$ or $\mathcal{O}^{\text{Leave}}$ but neither $\mathcal{O}^{\text{Corrupt}}$ nor $\mathcal{O}^{\text{Revoke}}$ is invoked. The output of each oracle query is defined as follows:

• For a session sid $= \tau$ $(l_3 + 1 \leqslant \tau \leqslant t - 1)$, the simulator invokes G which takes as input $s^{(\tau-1)}$ to output $(k^{(\tau)}, s^{(\tau)}) = G(s^{(\tau-1)})$. The output of public strings and secret keys for all key nodes and user nodes are exactly the same as that defined in π;

• For the session sid $= t$, the simulator invokes G which takes as input $s^{(t-1)}$ to output $(k^{(t)}, s^{(t)}) = G(s^{(t-1)})$ and then chooses a string $k'^{(t)} \in \{0,1\}^m$ uniformly at random; The simulator chooses a bit $b \in \{0,1\}$ uniformly at random. The leaf node secret key is assigned with the challenging string k. The selected $k_b \in \{k^{(t)}, k'^{(t)}\}$ is served as the group key and then it is encrypted as that specified in the protocol π.

All public strings and secret keys are computed according to the current user set $U^{(t)}$ and $(pk_{\text{GM}}^{(t)}, sk_{\text{GM}}^{(t)})$. The simulator outputs $U^{(t)}$ and $(pk_{\text{GM}}^{(t)}, sk_{\text{GM}}^{(t)})$. The adversary is $U^{(t)}$ and $pk_{\text{GM}}^{(t)}$.

• For each session sid $= \tau$ $(t + 1 \leqslant \tau \leqslant L - 1)$, the simulator invokes G which takes as input $s^{(\tau-1)}$ to output $(k^{(\tau)}, s^{(\tau)}) = G(s^{(\tau-1)})$. The output of public string and secret key for all key nodes and user nodes are exactly the same as that defined in π.

• Finally, $\mathcal{A}$ queries the Join oracle $\mathcal{O}^{\text{Join}}$ at session sid $= L$ and $\mathcal{A}$ is given $U^{(L)}$ and $(pk_{\text{GM}}^{(L)}, sk_{\mathcal{A}}^{(L)})$.

It is clear that G_0 and G_1 are identical assuming that the underlying symmetric-key encryption scheme is semantically secure. As such, the difference between G_0 and G_1 is at most d_{Enc} (the adversary's advantage defined in the semantic security game for the symmetric-key encryption scheme). We now analyze the adversary's advantage in the game G_1. If the challenging k is a random m-bit string, then no information about the bit b is leaked. If k is an output of the pseudo-random number generator, then G_0 and G_1 are identical. As a result, a PPT distinguisher D with the help of the adversary can be defined as follows: the input of D is a challenging string k, an adversary who is able to break the underlying multicast key distribution protocol π with non-negligible advantage ϵ, the output of D is what the adversary outputs in G_1. By the construction, we know that the advantage d_{PRG} of the distinguisher is bounded by $\epsilon + d_{\text{Enc}}$. As a result, if the underlying pseudo-random number generator is cryptographically secure, then the proposed protocol π achieves the backward security.

Similarly, we are able to prove the following lemma.

Lemma 2. The proposed multicast key distribution protocol achieves the forward-security in the presence of adaptive adversaries assuming that the underlying pseudo-random number generator is cryptographically strong and the symmetric-key encryption scheme is semantically secure.

Combining the two lemmas above together, we claim the following theorem.

Theorem 1. The proposed multicast key distribution protocol achieves the forward-security and the backward-security in the presence of adaptive adversaries assuming that the underlying pseudo-random

number generator is cryptographically strong and the symmetric-key encryption scheme is semantically secure.

4 Conclusion

In this paper, an efficient multicast key distribution system has been constructed from symmetric-key encryptions and pseudo-random number generators. The introduced multicast key tree decomposition technique allows a group manager to update internal states, generate and distribute public edge values and secret keys to the group users in an efficient manner. We have shown that the proposed multicast key distribution protocol achieves the forward-security and backward-security in the presence of adaptive adversaries assuming that the underlying pseudo-random number generator is cryptographically strong and the symmetric-key encryption scheme is semantically secure.

References

1 Sakarindr P, Ansari N. Survey of security services on group communications. IET Inf Secur, 2010, 4: 258–272
2 Burmester M, Desmedt Y. A secure and efficient conference key distribution system (extended abstract). In: Advances in Cryptology—EUROCRYPT'94. Berlin: Springer, 1995. 275–286
3 Kim Y, Perrig A, Tsudik G. Group key agreement efficient in communication. IEEE Trans Comput, 2004, 53: 905–921
4 Kim Y, Perrig A, Tsudik G. Tree-based group key agreement. ACM Trans Inf Syst Secur, 2014, 7: 60–96
5 Wu Q, Qin B, Zhang L, et al. Fast transmission to remote cooperative groups: a new key management paradigm. IEEE/ACM Trans Netw, 2013, 21: 621–633
6 Fiat A, Naor M. Broadcast encryption. In: Advances in Cryptology—CRYPTO'93. Berlin: Springer, 1993. 480–491
7 Boneh D, Gentry C, Waters B. Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Advances in Cryptology—CRYPTO. Berlin: Springer, 2005. 258–275
8 Gentry C, Waters B. Adaptive security in broadcast encryption systems (with short ciphertexts). In: Advances in Cryptology—EUROCRYPT. Berlin: Springer, 2009. 171–188
9 Phan D H, Pointcheval D, Shahandashti S F, et al. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. Int J Inf Sec, 2013, 12: 251–265
10 Wong C K, Gouda M G, Lam S S. Secure group communications using key graphs. In: Proceedings of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM, 1998. 68–79
11 Wong C K, Gouda M G, Lam S S. Secure group communications using key graphs. IEEE/ACM Trans Netw, 2000, 8: 16–30
12 Canetti R, Malkin T, Nissim K. Efficient communication-storage tradeoffs for multicast encryption. In: Advances in Cryptology—EUROCRYPT'99. Berlin: Springer, 1999. 459–474
13 Wallner D M, Harder E J, Agee R C. Key management for multicast: issues and architectures. National Security Agency, 1999. http://dx.doi.org/10.17487/RFC2627
14 Sherman A T, McGrew D A. Key establishment in large dynamic groups using one-way function trees. IEEE Trans Softw Eng, 2003, 29: 444–458
15 Goshi J, Ladner R E. Algorithms for dynamic multicast key distribution trees. In: Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing. New York: ACM, 2003. 243–251
16 Goodrich M T, Sun J Z, Tamassia R. Efficient tree-based revocation in groups of low-state devices. In: Advances in Cryptology—CRYPTO. Berlin: Springer, 2004. 511–527
17 Lysyanskaya A, Tamassia R, Triandopoulos N. Multicast authentication in fully adversarial networks. In: Proceedings of IEEE Symposium on Security and Privacy, Okaland, 2004. 241–255
18 Yao D, Fazio N, Dodis Y, et al. Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In: Proceedings of ACM Conference on Computer and Communications Security, Washington, 2004. 354–363
19 Zhu S, Setia S, Xu S, et al. Gkmpan: an efficient group rekeying scheme for secure multicast in ad-hoc networks. In: Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, 2004. 42–51
20 Xu S. On the security of group communication schemes. J Comput Secur, 2007, 15: 129–169
21 Chen Y R, Tygar J D, Tzeng W G. Secure group key management using uni-directional proxy re-encryption schemes. In: Proceedings of IEEE International Conference on Computer Communications, Shanghai, 2011. 1952–1960
22 Chen Y R, Tzeng W G. Efficient and provably-secure group key management scheme using key derivation. In: Proceedings of IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, 2012. 295–302
23 Cortier V, Steel G, Wiedling C. Revoke and let live: a secure key revocation api for cryptographic devices. In: Proceedings of ACM Conference on Computer and Communications Security, Raleigh, 2012: 918–928

24 Cho J H, Chan K S, Chen I R. Composite trust-based public key management in mobile ad hoc networks. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. New York: ACM, 2013. 1949–1956

25 Dong Q, Liu D, Ning P. Providing dos resistance for signature-based broadcast authentication in sensor networks. ACM Trans Embedded Comput Syst, 2013, 12: 73

26 Koskela T, Kassinen O, Harjula E, et al. P2P group management systems: a conceptual analysis. ACM Comput Surv, 2013, 45: 20

27 Kremer S, Künnemann R, Steel G. Universally composable key-management. In: Computer Security— ESORICS. Berlin: Springer, 2013. 327–344

28 Canetti R, Garay J A, Itkis G, et al. Multicast security: a taxonomy and some efficient constructions. In: Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, New York, 1999. 708–716

29 Micciancio D, Panjwani S. Corrupting one vs corrupting many: the case of broadcast and multicast encryption. In: Automata, Languages and Programming. Berlin: Springer, 2006. 70–82

30 Bellare M, Desai A, Pointcheval D, et al. Relations among notions of security for public-key encryption schemes. In: Advances in Cryptology—CRYPTO'98. Berlin: Springer, 1998. 26–45

31 National Institute of Standards and Technology. Announcing the Advanced Encryption Standard (AES): Federal Information Processing Standards Publication 197. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. 2001

32 Canetti R, Rivest R L, Sudan M, et al. Amplifying collision resistance: a complexity-theoretic treatment. In: Advances in Cryptology—CRYPTO. Berlin: Springer, 2007. 264–283

33 Aggarwal D, Dodis Y, Jafargholi Z, et al. Amplifying privacy in privacy amplification. In: Advances in Cryptology—CRYPTO. Berlin: Springer, 2014. 183–198

34 Dodis Y, Li X, Wooley T D, et al. Privacy amplification and nonmalleable extractors via character sums. SIAM J Comput, 2014, 43: 800–830

35 Halevi S, Harnik D, Pinkas B, et al. Proofs of ownership in remote storage systems. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. New York: ACM, 2011. 491–500

36 Dwork C, Naor M, Reingold O. Immunizing encryption schemes from decryption errors. In: Advances in Cryptology—EUROCRYPT. Berlin: Springer, 2004. 342–360

37 Goldreich O. The Foundations of Cryptography: Volume 2, Basic Applications. Cambridge: Cambridge University Press, 2004

38 Goldreich O. The Foundations of Cryptography: Volume 1, Basic Techniques. Cambridge: Cambridge University Press, 2001