A survey on optimization-based approaches to Dynamic Centralized Group Key management

Nguyen Thi Tuyet Trinh

Abstract—In secure multicast communication, only authorized group members are able to receive the transmitted data. Key generation, distribution, and management are all carried out by a single entity (called the Key Server) in the centralized system. The main challenges of a dynamic centralized group key management scheme include scalability overhead, rekeying overhead, storage overhead, maintaining forward and backward secrecy, etc. Logical Key Hierarchy (LKH), one of the key management methods, uses a tree structure to manage keys to share with participants. Due to the fact that members may join or exit the group at any time during communication, a crucial problem in centralized group key management (CGKM) is minimizing rekeying costs while maintaining a balanced tree. The majority of CGKM methods effectively update the group key by modifying cryptographic techniques or based on logical/heuristic arguments. One of the efficient approaches is based on an optimization model with the two mentioned important objectives. In this paper, we conduct a survey on optimization-based approaches to dynamic CGKM, which can be divided into two categories based on logical/heuristic arguments and a rigorous mathematical optimization model.

Keywords—Centralized group key management; Rekeying; DC Programming; DCA; combinatorial optimization.

I. INTRODUCTION

In the modern Internet era, multicast communication is widely used to transfer confidential information from a single sender to several receivers in a group. Multicast services, such as a video server transmitting networked TV channels, network games, pay-per-view, distance learning, and stock quotes, have played an important role in recent years due to the widespread use of multicast communication and the rapid growth of the Internet. As group membership is dynamic, it is required to update the group key in a safe and efficient manner when members join or leave the group.

Existing techniques to group key management can be categorized as centralized, decentralized, or distributed key management approaches. In centralized key management schemes, a single centralized server is responsible for generating and distributing group members’ keys. In decentralized key management methods, larger groups are separated into subgroups to reduce the computing workload on a centralized server and remove the single point of failure issue. On the other hand, distributed key management
techniques do not rely on a central server for the production and distribution of keys. In distributed key management methods, every member of the group participates in the generation and distribution of keys. Additionally, any member of the group may generate a shared group key from the key information obtained from the other group members. These three groups each have their own advantages and disadvantages, which are used in accordance with the requirements of applications in various fields.

In this paper, we conduct a survey on optimization-based approaches to dynamic centralized group key management (CGKM) using the widely-used hierarchical key-tree approach proposed by Wallner et al. [1] and independently studied by Wong et al. [2]. When there is a change in the group membership, the key server (KS) is typically responsible for updating the group key, which yields the security issues regarding forward and backward secrecy. The rekeying cost denotes the number of messages that needs to be distributed to the members so that they obtain the new group key. In the binary tree, only leaf nodes can be removed and new nodes will be only appended below given leaf nodes. The number of messages is the one of updated keys in the path from the root to the selected leaf node. By definition, a key tree is considered balanced if the distance from the root to any two leaf nodes differs by not more than one. As stated in [3], maintaining balanced trees is desirable in practice because membership updates can be performed with logarithmic rekeying costs provided that the tree is balanced. Hence, the tree's balance is a critical property of the LKH structure.

Various centralized key management approaches have been proposed in the literature for secure multicast communication. The majority of CGKM methods rely on the modification of cryptographic techniques or logical/heuristic arguments to update the group key efficiently. The optimization objectives include reducing the rekeying cost, the selection of the parameters in group communication systems, the structure of key tree, the time interval for rekeying, etc. However, most of these objectives have not been described systematically as a rigorous mathematical optimization model. Moreover, one of the efficient group key management approaches is based on an optimization model that takes into account simultaneously both objectives: the rekeying cost and the balance of the tree. Finding a set of leaf nodes in a binary key tree to insert new members and delete leaving members while minimizing the insertion cost and deletion cost constitutes a significant challenge in dynamic CGKM using LKH structure. At the same time, the approach needs to keep the key tree as balanced as possible. In fact, the key server cannot control the positions of departing members, the tree may become unbalanced afterward. It remains unbalanced until either insertions/deletions bring the tree back to a balanced state or some actions are taken to rebalance the tree. To overcome this problem, we need to control the shape of the key tree in the LKH. This objective should be carried out based on the insertion process since rebalancing the whole tree is a very expensive procedure.

The rest of the paper is organized as follows. Section 2 gives an overall description of hierarchical key-tree methods in CGKM. The survey on many relevant centralized key management optimization-based approaches for multicast communication is presented in Section 3, while Section 4 concludes the paper.

II. PRELIMINARIES OF CENTRALIZED GROUP KEY MANAGEMENT

Logical Key Hierarchy (LKH) [1, 2] is a common and fundamental technique for centralized group key management. The key tree is a hierarchical structure utilized by the LKH to manage the group key. A full binary tree in which each node has zero or two children is among the most efficient structures.

A. Terms and definitions

Before proceeding further, we introduce some notations and definitions [4] used in this paper.

- **Key**: sequence of symbols that controls the operations of a cryptographic transformation.

- **Individual key**: key shared between the key server and each member of the group.
- **Key encryption key**: cryptographic key that is used for the encryption or decryption of other keys.

- **Shared secret key**: key which is shared with all the active entities via a key establishment mechanism for multiple entities. It is also called group key.

- **Rekeying**: process of updating and redistributing the shared secret key (SSK), and optionally, key encryption key (KEK). This process is executed by the key server.

- **Individual rekeying**: rekeying method in which the shared secret key, and optionally, key encryption key are updated when an entity joins or leaves.

- **Batch rekeying**: rekeying method in which the shared secret key, and optionally, key encryption key are updated at every rekeying interval $T$.

- $e_K(M)$: result of encrypting data $M$ with a symmetric encryption algorithm using the secret key $K$.

- $X|Y$: result of concatenating data items $X$ and $Y$ in that order.

**B. Logical key hierarchy**

The LKH [1, 2], as illustrated in Figure 1, shows a single trusted key server to maintain the tree of keys and update the distribution of keys. A shared secret key is assigned to the root node of the tree. The leaf nodes of the key tree correspond to the group members and each leaf node assigns an individual key. Additionally, key encryption keys are assigned to the internal nodes (middle level nodes). The key encryption keys are shared by multiple members whose individual keys are assigned to the descendant of the node to which the key encryption key is assigned. Each member in the group has to maintain all the keys assigned to the nodes on the path from the root node to the leaf node, to which the individual key of the member is assigned. Therefore, the number of keys an entity has is proportional to the logarithm of the total number of entities. When a member joins or leaves, all the keys on the member's key path have to be changed to maintain forward and backward secrecy.

**Figure 1. A full binary tree structure**

Figure 1 shows an example of an LKH, it can be taken as a full binary tree architecture. Each node in a key tree represents a symmetric key. The key server chooses and distributes the keys for the eight members $U_1, U_2, ..., U_8$. The root key SSK represents the group key shared by all group members. The leaf node keys $IK_1, IK_2, ..., IK_8$ are assigned to the group members $U_1, U_2, ..., U_8$, separately, and $IK_i$ is shared by both the key server and individual member $U_i$. The internal node keys $KEK_1, KEK_2, ..., KEK_6$ are known only by the members that are in the subtree rooted at this node. Therefore, each member holds the keys from its leaf node key to the root node key. For instance, $U_1$ holds four keys $\{SSK, KEK_1, KEK_3, IK_1\}$ along the path from $U_1$ to root.

Defined a set of operations and techniques for efficiently updating the tree when a member join or leave the group [3].

**III. A SURVEY ON OPTIMIZATION-BASED APPROACHES TO DYNAMIC CENTRALIZED GROUP KEY MANAGEMENT**

In the centralized group key management system, a single organization is in charge of key generation, distribution, rekeying, and group communication. The greatest challenges of centralized methods are scalability overhead, rekeying overhead, communication overhead, computing overhead, storage overhead, maintaining forward and backward secrecy, and independence from collusion. The difficulty of centralized systems to manage multiple membership changes is an additional issue. To address these challenges, it is necessary to investigate the various key management optimization-based approaches described in the
For the establishment of keys in large, dynamic groups, the One-Way Function Tree (OFT) method was suggested in [11]. The computation cost of members is logarithmically dependent on the size of the group. The storage cost of group members is also proportional to the group's size on a logarithmic scale. However, the proposed scheme is vulnerable to collusion attacks, wherein departing and incoming members may collude their keying information to discover the previous and more recent group keys. The authors of [12] proposed three algorithms based on a 2–3 tree (where each internal node has a degree of 2 or 3) to reduce the worst-case communication cost required for updating the group key. They discovered that the height- and weight-balanced 2–3 tree algorithms achieved a reasonable trade-off between restructuring costs and tree structure. In order to balance the B-tree, a node split occurs after a new member is added. In [13], Lu et al. proposed a new Non Split Balancing High Order (NSBHO) tree in which the node splitting is not required for balancing the tree. To perform a leave operation, it requires the same worst case rekeying cost as 2–3 tree does. In addition, NSBHO has superior and far superior rekeying performance in average and worst case in comparison with 2–3 tree.

In [5], the authors demonstrated how to use a batching technique on the key server to process join and leave requests to decrease the rekeying cost. In [7], two merging methods that are appropriate for batch joining events to maintain the balance of the tree following insertion were developed. In [9], a multicast key management scheme was proposed for batch rekeying based on rotation one or twice. The balance factor of a node is the difference between the heights of its right and left subtrees. They reduced the overhead of batch rekeying operations when the batch leave operations are higher than batch join operations. The proposed rotation based algorithms has the ability to control the users join positions. However, they fail to control leaving positions of the users. The performance of the scheme is degraded, if the batch join operations are higher than the batch leave operations.

A. Optimization-based approaches based on logical/heuristic arguments

In the literature, there are several works that study the optimization aspect of CGKM with symmetric keys (keys that are shared secretly) [5-10]. As a solution to the scalability problem of group key management, the LKH key tree architecture was developed in [1, 2]. They implemented key graphs to identify secure groups. Additionally, they presented three strategies for the secure distribution of rekeying messages whenever the group membership changes. Nevertheless, the scheme’s computational cost increases. In [3], the authors addressed the problem of how to maintain balanced trees in the key management scheme of [1]. Every time a member is to be added in the tree, the key server always finds the shallowest leaf of the tree in which a new member to be inserted. Furthermore, they proposed two simple tree rebalancing schemes for deletion. One is a modification of the deletion algorithm that is to move a member located at the deepest level to the place occupied by the deleted member. The other method allows the key tree to be unbalanced for a while after a sequence of keys is updated, and then reconstructs a key tree to be in a balanced state. However, this method does not perform well when the leaving members are always on one side (right or left side). In this situation, the key tree becomes a skewed binary tree.
The key establishment mechanisms for multiple entities in order to provide procedures for managing cryptographic keying material used in symmetric or asymmetric cryptographic algorithms in accordance with the current security policy were specified in [4]. It also defines symmetric key establishment mechanisms based on a general tree structure (based on the LKH structure) with both individual and batch rekeying. This standard does not specify how to construct a tree structure or how to insert new members at specific positions in the tree.

Exclusion Basis Systems (EBS), a combinatorial formulation of the group key management problem that produces optimal results with respect to the size of the group, the number of keys stored by each member, and the number of rekeying messages was presented in [6]. They developed a general technique for determining the optimal values of the keys and messages as a function of the group size and described the tradeoff between the two mentioned parameters. Their formulation contains stars, $d$-ary trees and all such structures as special cases. Additionally, they illustrate the scalability of EBSs and provide an explanation of the algorithms for admitting and evicting group members. In [8], the authors suggested ways to optimize the key management structure in a hybrid group key management scheme. Their method is able to minimize both the total communication cost and the computational cost imposed on client devices. They proposed a hybrid group key management scheme in order to solve the problems associated with the star-based and tree-based schemes. The authors of [10] proposed a new batch rekeying scheme which dynamically configure the batch rekeying intervals. To analyze communication cost of a key tree structure, they utilize level-homogeneous key tree structure, whose node in the same level has the same degree.

Another way to solve the problem in group key management is to integrate the tree structure with the public key encryption algorithm. A Cluster-based Elliptic Curve Key Management (CECKM) protocol for secure group communications in wireless sensor networks (WSNs) was proposed in [14]. The CECKM protocol provides the same level of security as RSA and Diffie-Hellman with reduced keys and is effective for sensor node key exchange protocols. It requires significantly less time for system key resynchronization than the Diffie-Hellman and Group Diffie-Hellman protocols. In [15], the authors proposed a pairing-free Identity-based Two-Party Authenticated Key Agreement (ID-2PAKA) protocol using elliptic curve cryptography. The proposed protocol provides an efficient method for two parties to generate a shared session key over an open network and is suitable for efficient and secure peer-to-peer communications. In [16], Kumar et al. proposed a more efficient centralized group key distribution based upon RSA public key cryptosystem. In order to increase the level of the security, the proposed protocol uses an additional key called group key encryption key, which is generated using Chinese Remainder Theorem. In addition, the extended clustered tree structure based key management is more efficient for batch rekeying in terms of computational cost of KS. Finally, for updating the group key, whenever the group membership change, the amount of keying information that needs to be communicated is also reduced.

B. Optimization approaches based on an optimization model

It is noted that their above mentioned algorithmic designs are mainly based on logical/heuristic arguments. Broadly, the optimization objectives include reducing the rekeying cost, the selection of the parameters in group communication systems, the structure of key tree, the time interval for rekeying, etc. However, most of these objectives have not been described systematically as a rigorous mathematical optimization model. Consequently, the second research direction in dynamic CGKM is the development of an optimization model based on the structure of a binary key tree.

In [17], we developed an optimization approach to the problem of changing group keys in the LKH structure. We opt for the batch
inserting and individual deletion techniques. According to our knowledge, this is the first paper to introduce an optimization model that simultaneously considers both objectives: rekeying cost and tree's balance. As the new nodes to be inserted should be leaf nodes, the first step of our algorithm consists in finding the set of all leaf nodes of the given tree. In the second stage, based on the found leaf nodes, we consider an optimization problem which minimizes the insertion cost of new members while keeping the tree as balanced as possible. Clearly, the first step reduces considerably the complexity of the optimization model given in the second stage which works only on the leaf nodes. The optimization problem takes the following form:

\[
\min \sum_{i=1}^{l} (d_i - 1) \times \left| \sum_{j=1}^{m} x_{ij} \right|_0 \\
+ \lambda \left[ \max_{i=1,l} \log_2 \left( L[i] \times \left( \sum_{j=1}^{m} x_{ij} + 1 \right) \right) \right] \\
- \min_{i=1,l} \log_2 \left( L[i] \times \left( \sum_{j=1}^{m} x_{ij} + 1 \right) \right)
\]

subject to\[
\sum_{i=1}^{l} x_{ij} = 1, \forall j = 1,m, \forall \ i \in \{1,l\}, \forall \ j = 1,m,
\]

where \( \lambda \) is a positive parameter controlling the trade-off between the cost of inserting new members and the balance coefficient of the tree after insertion. It is observed that the optimization problem (1) is a very difficult optimization problem with nonsmooth, nonconvex objective function and binary variables. Due to the fact that the optimization model of the two-step algorithm is defined on the set of leaf nodes rather than the entire set of tree nodes, its space complexity is calculated as \( O(l) \).

In [18], we proposed a complete deterministic optimization model for solving the problem of updating group key in CGKM based on the entire set of tree nodes instead of only the set of leaf nodes. The proposed optimization model is a combinatorial problem with discontinuous step functions and binary variables.

\[
\min \sum_{i=1}^{n} (d_i - 1) \times \left| \sum_{j=1}^{m} x_{ij} \right|_0 \\
+ \lambda \left[ \max_{i=1,n} \left( T[i] \times B_i \times \left( \sum_{j=1}^{m} x_{ij} + 1 \right) \right) \right] \\
- \min_{i=1,n} \left( T[i] \times B_i \times \left( \sum_{j=1}^{m} x_{ij} + 1 \right) + (1 - B_i) \times C \right)
\]

subject to:

\[
\sum_{i=1}^{n} x_{ij} = 1, \forall j = 1,m, x_{ij} \in \{0,1\}, \forall \ i = 1,n, \forall \ j = 1,m.
\]

The total space complexity of this optimization model is \( O(n^2) \), where \( n \) is the number of tree nodes. It is very challenging to handle such kinds of programs by standard methods where the source of difficulty comes from the discontinuity of the objective and the binary nature of the solutions.

In [19], we proposed an optimization approach to the problem of batch deletion and insertion members in the LKH structure. Our primary objective is to simultaneously minimize the cost of removing and adding nodes while maintaining a balanced tree. In actuality, the deletion nodes are also the leaf nodes in the tree and a subtree of new nodes can be appended below a leaf node or is replaced the position of leaving node on the binary key tree. The latter has a lower updating key cost than the former since when a member leaves, all the keys on the path from the root to the deletion node must be updated anyway. The optimization model for the problem of updating group key with batch rekeying technique is following:

\[
\min \sum_{i=1}^{l_1} (d_i - 1) \times \left| \sum_{j=1}^{m} x_{ij} \right|_0 - \sum_{k=1}^{l_2} \sum_{j=1}^{m} y_{ij}
\]
subject to:

\[
\sum_{i=1}^{l_1} x_{ij} + \sum_{k=1}^{l_2} y_{kj} = 1, \forall j = 1, m, x_{ij} \in \{0,1\}, \forall i = 1, l_1, \forall k = 1, l_2, \forall j = 1, m.
\]

It is observed that (3) is an optimization problem with binary variables and discontinuous objective.

By introducing new binary variables and applying penalty techniques [20], the mentioned above problems (1-3) are first equivalently formulated to remove the discontinuity of the objective, and then the latter problem is reformulated as a DC (Difference of Convex functions) program via an exact penalty technique, where the DCA (DC Algorithm) is at our disposal as an efficient algorithm in DC programming [20]. DC programming and DCA, which constitute the backbone of nonconvex programming and global optimization, were introduced by Pham Dinh Tao in 1985 and have been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao. This theoretical and algorithmic framework has been applied successfully to various areas, namely, transport logistics, finance, data mining and machine learning, computational chemistry, computational biology, robotics and computer vision, combinatorial optimization, cryptology, inverse problems and ill-posed problems [21-25].

Overall, our proposed models provide a good compromise compared to existing works, producing a more balanced key tree with a low rekeying cost.

### IV. CONCLUSION

In this paper, we have reviewed some works on the optimization-based approaches to dynamic centralized group key management. An important problem in dynamic CGKM using LKH structure is to find a set of leaf nodes in a binary key tree to insert new members and delete departing members while taking into account simultaneously both objectives: the rekeying cost and the balance of the tree. In the literature, the majority of the logical/heuristic arguments used to support current algorithmic designs. Another way to solve the problem of group key updating when membership changes is based on an optimization model. In [17, 18], we introduced two optimization models to the problem of updating group key with batch insertion and individual deletion techniques based on the set of leaf nodes and tree nodes. In [19], we considered both the insertion cost and the deletion cost while keeping the tree as balanced as possible after batch rekeying. These are the first optimization models that considers simultaneously the updating key cost and the balance of the resulting key tree. The suggested optimization problems have binary variables and an objective function that is discontinuous. Using recent results on exact penalty techniques in DC programming, the problem can be reformulated as a DC program, for which the DCA is a useful algorithm. It has been shown that our approaches obtain a better trade-off between two considered criteria in comparison with existing approaches.
REFERENCES


**ABOUT THE AUTHOR**

**Nguyen Thi Tuyet Trinh**

Workplace: Université de Lorraine, LGIPM, Département IA, F-57000, Metz, France ; Vietnam Academy of Cryptography techniques.

Email: thi-tuyet-trinh.nguyen@univ-lorraine.fr

Education: Received the BSc degree from Beijing Jiaotong University in 2012; Received the MSc degree from Academy of Cryptography techniques in 2017.

Recent research interests: Optimization in cryptography.