A Tree Based Approach in Key Management Protocol for Multicast Communications

Sayed Mohammed Raja, Seetha. R
School of Information Technology, VIT University, Vellore – 632014, Tamil Nadu, India

ABSTRACT

Secure multicast transmission is a challenging issue in Ad-hoc networks due to its characteristics like high link failures and limited resources like power and storage. Hence managing of keys is considered as a vital role for the providing data correctness and confidentiality. Most popular group key management protocols of reliable multicast data communications used several algorithms to reduce the no of update messages during key updates. To overcome the scalability problem, proposed a new group key management scheme which uses m-ary Storage Efficient Tree (MSET) structure. Even though a tree structure decreases the communication cost, it requires more computing capability to perform crypto operations and needs more storage capacity to store various types of keys. Here we are restricting the no of nodes in a level to m. if leaf node decreases by m then the remaining nodes to be joined in the group. Here the maximum height of tree is log₂m (we divide the whole tree into two one as flat tree and the other as m-ary tree) only keys of m-ary tree are updated upon join or leaving of nodes is adjusted as binary tree. We are using immediate rekeying but it reduces the communication and storage costs. It reduces key update operations compared to earlier methods, by selecting some part of tree as flat tree and the keys of immediate upper level of users are updated. We use Boolean function minimization to reduce no of keys stored at group manager and group member (simulation and analysis using three cost metrics states that proposed protocol reduces storage, communication and computation capacity). Cost metrics we have used are number of keys stored, update messages exchanged between group members, and members of a group.

INTRODUCTION

Multicast communication is the essential technology which offers efficient services like IPTV, VOIP, and smart classrooms (Cisco Web-Ex) related to multimedia. In case of multicast communications data is protected usually implementing cryptography algorithms, thereby data confidentiality and integrity is guaranteed. In real time environment, data is protected by employing access control mechanisms. Here they use a Key Encryption Key(KEK) for data encryption and it is to be distributed to all authorized users so that they can decrypt the appropriate message[1]. Users who are having KEK and IK can perform decryption. Whenever a new member joins a group or current user leaves from a group, keys must be updated to provide
forward confidentiality as well as backward confidentiality and updated keys should be distributed to authorized members [2]. Backward confidentiality defines that, a member who recently joins a group must not have access to keys which are used in previous crypto operations, and forward confidentiality defines that, a member who left the group must not have access to future keys of the group. In this course of action, number of update messages depends on number of members in a group. If number of members in a group increases then there might be a scalability problem. To overcome this problem a hierarchal structure is implemented. Several tree based key management techniques have been projected to manage the key updating operations efficiently [3]. The proposed protocols key idea is to split group members as subgroups and a sub-group further separated as smaller sub-groups. But here number of update messages is increased exponentially. Thus the existing protocols suffer from scalability problem. To sustain the tree structure, group members should pile up more keys which affect the storage capacity of a mobile device.

In case of cryptographic techniques, a mobiles device consumes 30% more energy to decrypt than the energy used to encrypt the data [4]. Updated keys are also distributed in encrypted format; members of that group should perform decryption many times to obtain a new key. Thus to provide effective communication among the users with inadequate resources, cost of key computations, communication and storage cost must be reduced. As a result, performance evolution of the group key management protocol uses three metrics namely, computation, communication and storage cost. Though these three metrics are all crucial, the communication cost is the most considerable metrics while addressing the scalability problem in implementing a group based protocol for key management.

In this article, we propose a new hierarchal structure called m-ary tree and a tree management protocol called the M-ary Storage Efficient Tree (MSET) protocol which reduces the amount of keys by using Boolean function minimization, which obtains less storage, computation costs and overcomes the problem of scalability. The proposed key protocol removes the redundant tree levels which show little impudence on the cost of communication as the ratio of members leaving a group is quite large. By means of this representation, the costs of communication for different key hierarchal structures are examined, such as the M-ary tree; level based homogeneous tree arrangement. In addition, key tree arrangements are examined using different cost metrics such as communication, storage and computation, but most of the group key management techniques will considered only communication cost as most influential metrics. The crucial parts of this article can be summed up as follows: (i) We project a storage efficient key management scheme to maximize the effectiveness of storage and computation costs (ii) We design a new tree organization, m-ary tree (maximum of level 3) which allows a member to connect a multicast group securely, even though device may not have more memory and high processing power. (iii) We set a simple general key tree organization, called as level-
based homogeneous tree organization to grant an expression, its analysis of simple key tree organizations.

KEY TREE ORGANIZATION

In this part, we define a new key hierarchal structure, known as M-ary Storage-Efficient Key Tree (MSET), contrived to decrease the storage and computation costs. To begin the discussions, a concise overview about the typical tree organization is afforded first after that a new structure, called m-ary key tree structure, is projected. The proposed tree structures can be assorted as rendered in the below figure.

![Fig 1 Classification of key tree structure](image1)

TYPICAL KEY STRUCTURE

Typical key tree structure consists of different keys namely Individual Key (IK), subgroup key and group key. Group key is also known as Traffic Encryption Key (TEK). TEK is used by the node in the top most level of the tree. Sub-group key is also known as Key Encryption Key (KEK). KEK is used to generate a new key by encrypting the old keys if an user leaves or new user comes into the multicast group. Individual Key (IK) is distributed uniquely to all authorized members of a group. The following diagram shows a typical key tree structure having eight group members. It represents a binary tree, in which each parent node possesses two child nodes. In the below figure k is the TEK and $k_1, k_2... k_6$ are called the KEKs.

![Fig 2 A typical key tree structure](image2)
If the group key is changed and the previous key tree organization is not used then Key Distribution Center (KDC) have to transmit the updated keys in encrypted format like \( \{k'_{i=5, 6, 7 and 8}\} \) for particular group members (U8, U7, U6 and U5). Nevertheless, with a key tree structure, only one encryption message is transmitted by the KDC i.e. \( \{K_2\} \) to deliver a new group key for the sub-group members who are having key \( k_2 \).

**LEVEL-BASED HOMOGENEOUS TREE STRUCTURE**

Since a key tree organization facilitates the scalability problem in stepping up communication cost, researchers have undergone the schemes in terms of that which key tree structure can reduce the cost of communication [5]. Nevertheless, these articles provide a study of communication cost for binary key tree schemes in provisions of the number of key update messages. In binary key tree structure all intermediate nodes are possessed with two number of child nodes. Hence, the only possible way to notate binary key tree is, \( N = 2^H \). In this notation, \( H \) denotes the height of a tree. In case, if the number of members in a group is 65 or 113, it is impossible to find out the cost of communication. To analyze and express smoothly every member in a same level should have same degree. Therefore binary tree is a part of level-base homogenous structure.

**M-ARY STORAGE EFFICIENT TREE (MSET)**

In a tree based key management protocol, the key distribution center is responsible of encrypting the new keys with possible old keys and distributing them to specific members of group. Thus, to acquire new group key and sub-group keys, members of sub-group should perform decryption on new keys using their old keys. KDC will not distribute any sort of keys to newly joined members. The process of obtaining new keys includes lots of operations, such as decryption, set the KDC to offer delay in obtaining the new group key for the data encryption of a communication system, nodes need to consume more power to acquire the new TEK and KEK. Hence, the number of operation like encryptions, decryptions should be reduced in a efficient group key management protocol. Nevertheless, many keys like individual, sub-group and group keys should be stored at a communication device as well as at group members. It should not be stored in a common memory because, a non-legitimate user can decrypt the data using that keys. So, they are stored in a secure memory called as Universal Subscriber Identity Module (USIM). Every computation requires memory to store their secure keys as well as secure data. Amount of Secured memory space provided to a mobile device is very limited; we should minimize the number of keys to be stored to perform computations. If the level of tree is minimized, automatically the number of keys to be stored will be reduced. In view of all the above stated constraints, we propose the MSET. The MSET consists of two parts, where the bottom part of the MSET consists of binary trees which helps in reducing the communication cost, and the top most part of the MSET is a flat tree structure which helps in minimizing the storage and
computation costs. The group key is directly accessed by the sub-group heads and the sub-group keys (KEK) are generated based on the keys of group members. So that it is a part of the level-based homogeneous tree structure, as shown in above figure, and denoted as $T(m, m, \ldots m^{h-k})$ where $h=\log_2[n]$ indicates binary tree (complete) and $n$ denotes number of group members.

![Proposed protocol M-ary storage efficient key hierarchy of level 2](image)

**MEMBER LEAVING AND KEY UPDATE OPERATION**

The proposed protocol uses Boolean Function Minimization (BFM) technique which reduces the size of key to be stored at each member. It is the best solutions proposed so far to solve the problem of scalability in multicast key management protocol which depends up on a centralized authority. Rather than the usage of one way tree as in BFM, the members are classified into more number of sub-group trees [6]. If more than one member left from the group in same round, it uses aggregate deletion of members from the group rather than deleting one by one from the group. If we delete one by one from a group, after every deletion operation keys are to be updated. Consider a set of members $Ui=\{U1, U2, \ldots , Un\}$ where $N=ncy \times 2^y$ each user has $y$ auxiliary keys with their respective combinations. An illustration form the above figure U9 knows IK9, KEK2, IK10, IK9 IK10, IK9 IK'10. The new TEK can be obtained by encrypting the
key if all the auxiliary keys are voided due to removal of members from the group. To update KEK, we use a pattern which consists of single auxiliary key to address all the remaining members of a sub-group. If single auxiliary key pattern fails to address all the remaining members of sub-group, we use double auxiliary keys in patterns and so on until remaining members are covered. Same procedure is followed when a new member joins the group.

RESULTS

<table>
<thead>
<tr>
<th>n</th>
<th>y</th>
<th>N</th>
<th>2^n</th>
<th>m</th>
<th>2^m</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>32</td>
<td>16</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>15360</td>
<td>1024</td>
<td>15</td>
<td>16384</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>3,075,072</td>
<td>32,768</td>
<td>23</td>
<td>4,194,304</td>
</tr>
<tr>
<td>20</td>
<td>17</td>
<td>1,4176e+011</td>
<td>33,554,432</td>
<td>39</td>
<td>2,7488e+11</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>3,1504e+013</td>
<td>1,0738e+09</td>
<td>42</td>
<td>3,5184e+13</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>1,615e+018</td>
<td>1,0995e+12</td>
<td>60</td>
<td>2,3058e+18</td>
</tr>
<tr>
<td>50</td>
<td>31</td>
<td>3,7030e+020</td>
<td>4,5184e+13</td>
<td>70</td>
<td>5,9030e+20</td>
</tr>
</tbody>
</table>

Table. Comparison of proposed protocol with MSET

n: Auxiliary keys pairs stored at a group manager in MSET
y: Auxiliary keys stored at each group member in MSET.
N: Number of members in a group.
2^n: Members of group in MSET protocol having the same storage capacity as a group manager.
m: Auxiliary key pairs at group manager in MSET protocol in order to address the same group

Fig .4 key storage comparison at Group manager  Fig .5 Update messages at Group manager
COMPARISON BETWEEN CENTRALIZED KEY DISTRIBUTION TECHNIQUES

We, first compare the centralized key distribution protocols: OFT, LKH and MSET techniques. Since, MSET protocol experiences the least communication overhead and less storage requirements when compared to other protocols as shown below. Hence, a comparison between LKH, OFT and M-ary storage efficient tree protocol is detailed. In order to compare the above stated centralized protocols, some of the following general assumptions are taken into consideration:

i) Number of members in a group $N_g$.
ii) Height of a tree $h$, where $h=\log_2(N_g)$.
iii) Time taken to encrypt and decrypt.
iv) The length of re-key message which we use to update the group key.
v) Storage capacity required at sub-group managers and group members.

<table>
<thead>
<tr>
<th></th>
<th>LKH</th>
<th>OFT</th>
<th>MSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward secrecy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Backward secrecy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Member join: encryption operation</td>
<td>$2h$ keys</td>
<td>$(3h/2)+2$ keys</td>
<td>$(3h/2)+2$ keys</td>
</tr>
<tr>
<td>Rekey message length</td>
<td>$2h$</td>
<td>$(h/4)+1$</td>
<td>$(h/4)+1$</td>
</tr>
<tr>
<td>Member leave: encryption operations</td>
<td>$h(h-1)/2$</td>
<td>$3h/2$</td>
<td>$3h/2$</td>
</tr>
<tr>
<td>Re-key message length</td>
<td>$h(h-1)/2$</td>
<td>$h$</td>
<td>$h$</td>
</tr>
<tr>
<td>Storage requirements</td>
<td>$2N_g-1$ keys</td>
<td>$2N_g-1$ keys</td>
<td>$2h$ keys</td>
</tr>
</tbody>
</table>

Table 1 Comparison of proposed protocol with earlier key management protocols

The proposed protocol requires less storage capacity at both the sub-group manager and group members as shown in above graphs. Therefore, this satisfies all requirements of secure multicast
key management protocol as stated in [7]. In conclusion, the proposed protocol achieves less storage cost at both user level and group level. On the other hand, the proposed protocol provides less communication overhead if a single user leaves and a comparable overhead in case of numerous leaves in same round.

CONCLUSION

In this paper, we have discussed the problems of managing the keys in multicast communication. As a solution to the above discussed problems we have given a new key tree management protocol called MSET, which uses level homogenous tree and a tree based key management protocol which dynamically controls the structure of key tree, considering the storage, communication and computation cost. By handling key tree structure the MSET smoothly minimizes the storage and computation cost. It uses Boolean function minimization to reduce the length of keys at every level. We also analyzed the proposed protocol by adding more number of members to a group and deleting from a group. We have considered update messages, length of key and number of members as influential factors and performed simulations. Based on the statistics we have shown that a proposed protocol reduces the computation and storage costs to a greater extent.

REFERENCES


