Group-Key Generation through Ternary Tree Based Rebuild Algorithm

V.Srinadh, P.V. Nageswara Rao

Abstract: Security plays essential role in any correspondence framework particularly in Group oriented correspondence. In Group oriented correspondence, entire correspondence will occur with the help of one secret key which is called Group Key. This gathering key must be changed at whatever point another part joins into the gathering just as a current part leaves from the gathering; which is known as rekeying. This gathering correspondence can be spoken to utilizing key tree. In case we utilize ternary tree, totalness of the tree will be expanded if the group members are more, so that rekeying activity takes additional time. Rather than ternary tree on the off chance that we use quad tree, so that the tree stature will be less, so that rekeying activity takes less time. So the correspondence will be increasingly secure and quicker. In this paper we are going to implement Ternary tree based Rebuild Algorithm.

Key-Words: - Group key, Rekeying, security, Ternary tree, Rebuild Algorithm

Key-Words: - Group key, Rekeying, backward confidentiality, forward confidentiality, security, Ternary tree

1. INTRODUCTION

Security assumes imperative job in any correspondence framework particularly in Group situated correspondence. In gathering focused correspondence framework whole correspondence will happen with the assistance of one mystery key which is called Group Key. This gathering key must be changed at whatever point another part joins into the gathering. With the goal that the joined individuals can't get to the past imparted information. Just as gathering key must be adjusted at whatever point a current part leaves from the gathering with the goal that the left individuals can't get to the further correspondence information. Changing the gathering key at whatever point another part joins into the gathering and a current part leaves from the gathering is known as rekeying. This gathering correspondence can be spoken to utilizing key tree. On the off chance that we utilize Ternary tree, totalness of the tree will be expanded if the individuals are more, so that rekeying activity takes additional time. Rather than Ternary tree on the off chance that we utilize quad tree, so that the tree stature will be less, so that rekeying activity takes less time. To accomplish this we need a protected dispersed gathering key understanding and validation convention with the goal that individuals can set up and confirm a typical gathering key for secure and private correspondence. First key understanding convention was proposed by Diffie-Hellman. It can ensure the security of correspondence between the two clients. Tree-based Group Diffie-Hellman (TGDH) is one of the conventions that stretch out the Diffie-Hellman convention to a gathering key understanding protocol [1].

II. TREE-BASED GROUP DIFFIE–HELLMAN PROTOCOL

To efficiently maintain the group key in a dynamic peer group with more than two members, we use the tree-based group Diffie–Hellman (TGDH) protocol proposed in [10]. Each member maintains a set of keys, which are arranged in a hierarchical binary tree. We assign a node ID $v$ to every tree node. For a given node $v$, we associate a secret (or private) Key $K_v$ and a blinded (or public) key $BK_v$. All arithmetic operations are performed in a cyclic group of prime order $p$ with the generator $\alpha$. Therefore, the blinded key of node can be generated by

\[ BK_v = \alpha^{K_v} \mod p \quad \ldots \ldots (1) \]

Each leaf node in the tree corresponds to the individual secret and blinded keys of a group member $M_i$. Every member holds all the secret keys along its key path starting from its associated leaf node up to the root node. Therefore, the secret key held by the root node is shared by all the members and is regarded as the group key $K_1$.

The node ID of the root node is set to 0. Each non leaf node consists of either three child nodes whose node ID’s are given by $3v+1, 3v+2$ and $3v+3$ for its left child, middle child and right child respectively or two child nodes whose node ID’s are given by $3v+1$ and $3v+2$ for its left child and middle child respectively. Based on the Diffie–Hellman protocol [3], the secret key of a non leaf node can be generated, in two cases, as follows:

**case 1: Non leaf node with two child nodes**

Let $v$ be the non leaf node whose two children are $3v+1$ and $3v+2$ then the secret key of $v$ can be calculated by the secret key of one child node of $v$ and blinded key of another child node of $v$. Mathematically, we have

\[ K_v = (BK_{3v+1}K_{3v+2})^{K_v} \mod P \]

\[ = (BK_{3v+1})^{K_v} \mod P \quad \ldots \ldots (2) \]

\[ = \alpha^{K_{3v+1}K_{3v+2}} \mod P \]
case 2: Non leaf node with three child nodes

Let v be the non leaf node whose three children are 3v+1, 3v+2 and 3v+2c then the secret key of v can be calculated by the secret key of one child node of v and blinded key of other two child nodes of v. Mathematically, we have

$$K_v = (BK_{3v+3})^((BK_{3v+3})K_{3v+1}) \mod P \mod P$$

$$K_v = (BK_{3v+2})^((BK_{3v+2})K_{3v+1}) \mod P \mod P$$

$$K_v = (BK_{3v+1})^((BK_{3v+1})K_{3v+2}) \mod P \mod P$$

$$K_v = (BK_{3v+3})^((BK_{3v+3})K_{3v+2}) \mod P \mod P$$

$$K_v = (BK_{3v+2})^((BK_{3v+2})K_{3v+3}) \mod P \mod P$$

$$K_v = (BK_{3v+1})^((BK_{3v+1})K_{3v+3}) \mod P \mod P$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldot
At the beginning of each rekeying interim if the sum of current members stay in the communication group and the members who are recently joining the group is equal to four then the key tree will be constructed using Rebuild algorithm as shown in fig.4. If this number is equal to five then the key tree will be constructed using Rebuild algorithm as shown in fig.5. If this number is equal to six then the key tree will be constructed using Rebuild algorithm as shown in fig.6.

Fig. 3. Key tree with three members in Rebuild

At the beginning of each rekeying interim if the sum of current members stay in the communication group and the members who are recently joining the group is equal to four then the key tree will be constructed using Rebuild algorithm as shown in fig.4. If this number is equal to five then the key tree will be constructed using Rebuild algorithm as shown in fig.5. If this number is equal to six then the key tree will be constructed using Rebuild algorithm as shown in fig.6.

Fig. 4. Key tree with four members in Rebuild

Fig. 5. Key tree with five members in Rebuild

IV. REKEYING

Rekeying (renewing the keys of the nodes) is performed for every single (multiple) join / leave event to ensure backward and forward confidentiality. A special member called sponsor is elected to be responsible for broadcasting updated blinded keys. Consider the key tree with 12 members (M1 to M12) as shown in figure 7. Consider the scenario if M2, M4, M5, M7, M9 and M12 leave the group while M13, M14, M15 and M16 join the group. Then Ternary tree based rebuild algorithm works as follows: First, identify the list of members available in the group. Here the group is having 12 members M1 to M12. Unlike individual rekeying, Ternary tree based Rebuild Algorithm will perform rekeying one time per one set of leaves and joins simultaneously. Now remove the leaving nodes i.e. remove M2, M4, M5, M7, M9 and M12 from the list and add the joining nodes M13, M14, M15 and M16 to the list simultaneously. So the new list is with group members M1, M3, M6, M8, M10, M11, M13, M14, M15 and M16

Fig. 6. Key tree with six members in Rebuild

As the new list is having 10 members the Ternary tree based Rebuild algorithm constructs the key tree as shown in fig.8. Now the group key has to be changed. The rekeying operations: M13, M14, M15, and M16 broadcast their blinded keys on their joining. M16 is one of the sponsors. M16 computes K3 (SG3) using its private key K12 along with M14’s blinded key BK10 and M15’s blinded key BK11. M16 broadcasts BK3. M13 is one of the sponsors. M13 computes K2 (SG2) using its private key K9 along with M10’s blinded key BK7 and M11’s blinded key BK8. M13 broadcasts BK2. M3 will become one of the sponsors. M3 computes secret key K1 using K4, M6’s blinded key BK5 and M8’s blinded key BK6. M3 broadcasts BK1.
BK1 and BK2, M16 can compute K0 which is group key. The same way remaining members can
compute secure group key. Communication will happen as long as there is no leave event or join event or both. When there is a member wants to join the group or leave the group then the group-key will be changed and a similar way it will be processed.

Fig.8. Key tree using Rebuild after M2, M4, M5, M7, M9 and M12 leave the group while M13, M14, M15, and M16 join the group

V. MATHEMATICAL ANALYSIS

Consider a group with N members and let L be the number of members leaving the group and J be the number of members joining the group. Now the resultant group members N* can be calculated as follows:

\[ N^* = N - L + J \]  (4)

Table 1. Height of the binary tree and ternary tree for a different number of members of the group.

<table>
<thead>
<tr>
<th>No. Of Nodes</th>
<th>Height of the Binary tree</th>
<th>Height of the Ternary tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.63</td>
</tr>
<tr>
<td>3</td>
<td>1.58</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1.26</td>
</tr>
<tr>
<td>5</td>
<td>2.32</td>
<td>1.46</td>
</tr>
<tr>
<td>6</td>
<td>2.58</td>
<td>1.63</td>
</tr>
<tr>
<td>7</td>
<td>2.81</td>
<td>1.77</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1.89</td>
</tr>
<tr>
<td>9</td>
<td>3.17</td>
<td>2</td>
</tr>
</tbody>
</table>

Height of the binary tree and ternary tree for a different number of members of the group can be represented as shown in the Table 1. The tree height comparison graph is as shown in fig.9.
VI. CONCLUSION

To represent group members if we use binary tree, height of the tree will be increased if the members are more, so that rekeying operation takes more time. Instead of Binary tree if we use ternary tree the tree height is less compared to binary tree, so that rekeying operation takes less time. So the communication will be more secure and faster.

REFERENCES:


AUTHORS PROFILE

Srinadh.Vemireddi received the Ph.Degree in Computer Science from the University of GITAM University, Visakhapatnam in March, 2019. He is currently an Assistant professor of Computer Science and Engineering at GMRIT, Rajam, Andhra Pradesh.

P.V. Nageswara Rao received M.Tech (CST) from Andhra University and Ph.D from Nagarjuna University and had 23 years of teaching and research experience. Presently working as Professor, Department of CSE, GITAM University, Visakhapatnam. Research interests include Bioinformatics, Soft Computing and Data Mining.