Storage and Rekeying Cost for Cumulative Member Removal in Secure Group Communication

A S Poornima[1], R. Aparna [1], Dr. B.B. Amberker [2]

[1]Dept. of Computer Science & Engineering Siddaganga Institute of Technology
Tumkur 572 103, Karnataka, India
[2]Dept. of Computer Science & EngineeringNIT, Warangal, A.P. India

Summary
Many applications like pay-per-view, distribution of digital media etc., require secure group communication services in order to deliver packets from one or more authorized senders to a large number of authorized receivers. The main issue in secure group communication is group dynamics and key management. A scalable secure group communication model ensures that whenever there is a membership change, new group key is computed and distributed to the group members with minimal computation and communication cost. Handling member removal(leave) is more complex than member join event in any secure group communication model. In this paper m-ary tree structure is used, with number of keys at each level being m. Here, we address cumulative member removal(leave) and present protocols that minimize the number of messages required to distribute new group key to remaining members in the group. The issues related to two members removal(leave) and cumulative arbitrary members removal are handled separately.

Key words:
Cumulative member removal, encryption keys, secure group communication

1. Introduction

Applications such as pay-per-view, distribution of digital media, pay-per-use multi-party games, and restricted conferences fall in the category where the receiver set needs to be restricted to legitimate subscribers. To setup such a secure group, each secure multicast group is associated with one or more trusted servers responsible for managing membership to the group called as key server. When a client wants to join the group, the client and key server mutually authenticate using an authentication protocol. If the client is permitted to join the group, the key server provides it with the required keys. The keys sent to the client include the group key which is shared by all members of the group and auxiliary keys, depending upon the key distribution algorithm.

Manuscript received September 5, 2007
Manuscript revised September 20, 2007

The key server is also responsible for handling client removal and leaving event. Leaving is initiated by a client and is important in applications such as pay-per-view where a client leaving a group would like to ensure that it is no longer charged for usage. Removal of group member is usually initiated by a key server and is important in cases where the particular group member loses the access control privileges.

To prevent a new user (join operation) from reading past communications (backward access control) and a departed user (leave operation or removal) from reading future communications (forward access control), the key server has to change the group key (rekey operation) whenever group membership changes. For large groups, join and leave requests can happen frequently. Therefore a group key management service should be scalable with respect to frequent key changes.

The topic of key management for multiparty communications are studied in [1,4,5,6,7,8]. The scalability problem associated with frequent key changes in a large group is addressed in [10,2]. In [10] Iolus addresses the scalability problem by dividing a large group into multiple subgroups and employing a hierarchy of group security agents. The scheme proposed in [2] uses a hierarchy of keys to solve the scalability problem. A key update in this scheme requires O(log2 N) messages where N is the size of the group. In this scheme each client has to store log2 N keys (i.e., keys along the path from leaf to the root) and the key server has to maintain a tree of O(N) keys. The scheme proposed in [3] focuses on the problem of cumulative member removal and finds out the minimum number of messages required to distribute new keys to the remaining group members. The entire operation in this paper focuses on binary tree and uses Boolean Function Minimization techniques.

Our approach proposed in this paper also focuses on cumulative member removal with minimum rekey

messages to update the keys whenever there is a membership change. The scheme discussed in [3] is extended in this paper to m-ary tree instead of binary tree. Our scheme distributes new group key to the remaining group members with minimum number of messages as compared to the scheme in [2]. In our scheme, in order to avoid the leaving members using auxiliary keys to learn the new group key, auxiliary keys are also updated.

The paper is organized as follows: In Section 2 we discuss about motivation. In Section 3, we brief about the model and notations used in the paper, sections 4, 5 and 6 discuss about single member removal, two members removal and cumulative arbitrary members removal respectively along with protocols and comparison with Wong et al scheme [2].

2. Motivation

In [3] binary tree structure is used. When the group is large, the number of levels in the binary tree will be more which increases number of keys at user. Extending the scheme to *m*-ary tree will reduce the height of the tree reducing number of keys at each user. At the same time we should consider server side storage i.e., number of keys at every level of the key tree. In [3] two keys are maintained at every level of the key tree, extending the scheme to *m*-ary tree will result in maintaining *m* keys.

For a group size *n*, if *d* is the height of the binary tree, it results in storing 2**d* keys at the server. For the same value of *n*, if *d'* is the height of the *m*-ary tree, then *m***d'* keys are to be stored at the server. We can have the relation

$n = 2^d = m^{d'}$

$\rightarrow \quad d' = d/\log_2 m$

Number of keys at server in *m*-ary tree in terms of *d* can be represented as $m*(d/\log_2 m)$, which illustrates that as *m* increases, number of keys at server will increase, which violates our motto. Hence in order to maintain minimum number of keys both at user and server, following relation has to be satisfied :

$(m*d/\log_2 m) \leq 2*d$ which is true only if $m \leq 4$.

3. Model and Notations

m-ary tree: is a tree with the following properties:

(i) each interior node has at most *m* children

(ii) each path from the root to a leaf has the same length

N: Total number of users associated with the group and all users must be at the leaf level. Each user is assigned with Unique Identification Number (UID) which is a binary string of length ***n*** (where $n = \log_2 N$).

Subgroups: Each interior node containing at the maximum *m* children nodes forms one subgroup. Subgroups at level i are assigned with keys K_{i0} to $K_{i(m-1)}$ called Auxiliary keys at level i.

Keys: Individual user keys of any subgroup are numbered from K_0 to K_{m-1} so that all users at position 0 of all subgroups are assigned with key K_0 and all users at position 1 of all subgroups are assigned with key K_1 and so on up to K_{m-1}.

$\{GK\}_{K0}$ ⟶ denotes GK is encrypted with the key K_0.

$\|$ ⟶ denotes concatenation operation

From fig. 1 the values of N, *m*, *n*, keys, auxiliary keys and group key are as follows:

N=16 *m*=4 *n*=4

Keys:

Users u_0, u_4, u_8, u_{12} are assigned with key K_0
Users u_1, u_5, u_9, u_{13} are assigned with key K_1
Users u_2, u_6, u_{10}, u_{14} are assigned with key K_2
Users u_3, u_7, u_{11}, u_{15} are assigned with key K_3

$K_{10}, K_{11}, K_{12}, K_{13}$ are auxiliary keys at level 1.

GK is the group key shared by $u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}, u_{11}, u_{12}, u_{13}, u_{14}, u_{15}$.

4. Single member Removal (Leave)

If a single member wants to leave the multicast group voluntarily or is removed (expelled) from the group, a new group key must be computed and distributed to the remaining members in the group, so that leaving member will not be able to decrypt the future messages. The rekeying method used when a single member leaves the group is similar to the one used in [2]. In fig.1 if user u_2 leaves, the rekey message to distribute new group key, GK' is

$[\ \{GK'\}_{K0} \| \{GK'\}_{K1} \| \{GK'\}_{K3} \| \{GK'\}_{K11} \| \{GK'\}_{K12} \| \{GK'\}_{K13}\ ]$

After distributing new group key to remaining members in the multicast group securely, auxiliary keys are updated using the function F as follows:
F(auxiliary key, new group key) ← (Auxiliary key) ***XOR*** (New Group key)
Same method holds good for all the following cases to compute new auxiliary keys.

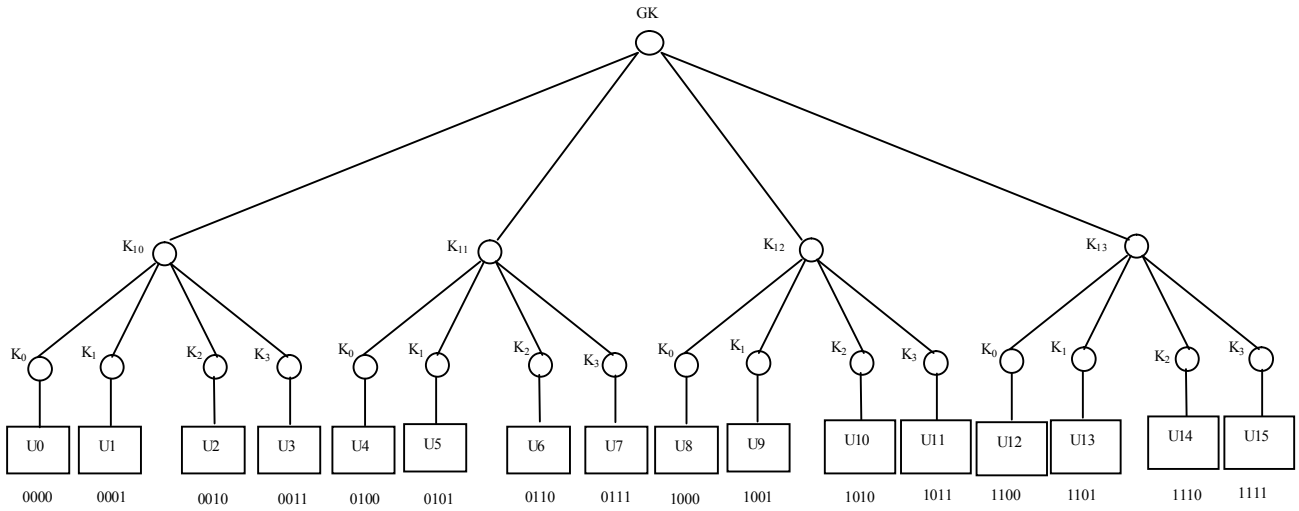

Fig.1. Key tree structure showing UIDs and keys of users in the group, auxiliary keys and group key

5. Two members removal (leave) at a time

When two members leave the multicast group voluntarily or being removed from the group, we need to address three different cases: (i) both the leaving members are from the same subgroup, (ii) leaving members belonging to different subgroups but at the same position with common individual key (for eg., in fig.1 u_1 and u_5 are the members belonging to different subgroups sharing the key K_1), (iii) leaving members belonging to different subgroups and also at different positions with different individual keys (for e.g., in fig.1 users u_4 and u_9 belong to subgroup 1 and 2 respectively with individual keys being K_0 and K_1 respectively).

Protocol 1 depicts the computation of encryption keys for two members removal (leave) case.

Notations used in Protocol 1:
Let L_1 and L_2 be the UIDs of leaving members.

KEK : is the set, initially empty, and at the end contains the keys used to encrypt the new group key.
P: Lower order $\log_2 m$ bits of L_1
Q: Lower order $\log_2 m$ bits of L_2
h_1 : Higher order $\log_2 m$ bits of L_1
h_2 : Higher order $\log_2 m$ bits of L_2
U : denotes set union operation

At the end of protocol 1, KEK contains the keys which are individually used to encrypt GK'.

Protocol 1 can be summarized for all the three cases using fig.1 as follows:

Case (i): let $L_1 = u_5$ and $L_2 = u_6$
/* leaving members from the same subgroup */

KEK = { K_{10}, K_{12} , K_{13}, K_0, K_3}

Following users can decrypt the new group key GK' encrypted using the keys of set KEK:

u_0, u_1, u_2, u_3	(using key K_{10})
u_8, u_9 ,u_{10}, u_{11}	(using key K_{12})
u_{12}, u_{13}, u_{14} , u_{15}	(using key K_{13})
u_4	(using key K_0)
u_7	(using key K_3)

For the same members removal, Wong et al. scheme of [2] requires 6 encryptions, whereas our scheme requires 5 encryptions.

```
Step 1:  Repeat thru Step 3
            for i ← 0 to m-1 do
                if (h1 ≠ i and h2 ≠ i )      /* leaving members not belonging to subgroup i*/
                   KEK← KEK U { key at the parent of subgroup i }
                Else
                  Goto Step 2.

Step 2: if ( P = =Q )     /* if leaving members are from the same position of different subgroups */
            Begin
                Repeat for j ← 0 to m-1 do
                       if ( j ≠ P )
                          KEK← KEK U { key at member j of subgroup i }
                Continue with Step 1
            End
         Else
            Goto Step 3

Step 3: /* if leaving members are from different positions of two different subgroups */
              Repeat for k ← 0 to m-1 do
                Begin
                    if ((k  ≠ P) and (k ≠ Q))
                          KEK← KEK U { key at member k of subgroup i }
                    else
                        if(k= =P) /* compute new encryption key  EK*/
                                EK = ( key at parent of member L2 ) XOR
                                      ( key at member k of subgroup i )
                                KEK← KEK U {EK}
                    else    /* k ≠ P but k = Q , compute new encryption key  EK*/
                                EK = ( key at parent of member L1 ) XOR
                                      ( key at member k of subgroup i )
                                KEK← KEK U {EK}
                 End

Step 4: return ( KEK )
```

Protocol 1 : Computation of encryption keys for two members removal (leave)

Case (ii): let $L_1 = u_1$ and $L_2 = u_9$
/* leaving members are from the same position of different subgroups */

KEK = { K_0, K_2 , K_3, K_{11}, K_{13}}

Following users can decrypt the new group key GK' encrypted using the keys of set KEK:

u_4, u_5, u_6, u_7	(using key K_{11})
u_{12}, u_{13} ,u_{14}, u_{15}	(using key K_{13})
u_0, u_8, u_4 , u_{12}	(using key K_0)
u_2, u_6, u_{10}, u_{14}	(using key K_2)
u_3, u_7 ,u_{11}, u_{15}	(using key K_3)

For the same members removal, Wong et al. scheme of [2] requires 10 encryptions, where as our scheme requires 5 encryptions.

Case (iii): let $L_1 = u_2$ and $L_2 = u_{13}$
/* leaving members are from different positions of two different subgroups */

KEK = { K_0, K_3, K_{11} , K_{12}, K_{13} ***XOR*** K_2 , K_{10} ***XOR*** K_1}

Following users can decrypt the new group key GK' encrypted using the keys of set KEK:

u_0, u_4, u_8, u_{12} (using key K_0)
u_3, u_7, u_{11}, u_{15} (using key K_3)
u_4, u_5, u_6, u_7 (using key K_{11})
u_8, u_9, u_{10}, u_{11} (using key K_{12})
u_{14} (using key K_{13} ***XOR*** K_2)
u_1 (using key K_{10} ***XOR*** K_1)

For the same members removal, Wong et al. scheme of [2] requires 10 encryptions, where as our scheme requires 6 encryptions.

6. Cumulative removal of Arbitrary members

Any number of members can leave (be removed from) the multicast group from any position in the *m*-ary tree. Protocol 2 handles the computation of encryption keys for cumulative removal of arbitrary members.

Let the number of leaving members be L. H is an array with L elements containing higher order $\log_2 m$ bits of leaving members. P is an array with L elements containing lower order $\log_2 m$ bits of leaving members.

KEK: is the set, initially empty, and at the end contains the keys used to encrypt the new group key.

S : User set, initially contains all the members in the multicast group excluding leaving members.

U: denotes set union operation.

Let leaving members be u_1, u_2, u_9, u_{15}

KEK = { K_0, K_{11}, K_1***XOR*** K_{13}, K_3***XOR*** K_{12},
K_3***XOR*** K_{10}, K_2***XOR*** K_{12},
K_2***XOR*** K_{13} }

Following users can decrypt the new group key GK' encrypted using the keys of set KEK:

u_4, u_5, u_6, u_7 (using key K_{11})
u_0, u_8, u_{12} (using key K_0)
u_{13} (using key K_1***XOR*** K_{13})
u_{11} (using key K_3***XOR*** K_{12})
u_3 (using key K_3***XOR*** K_{10})
u_{10} (using key K_2***XOR*** K_{12})
u_{14} (using key K_2***XOR*** K_{13})

For the same members removal, Wong et al. scheme of [2] requires 13 encryptions, where as our scheme requires 7 encryptions.

```
Step 1: /* for the subgroup in which no member is leaving  */
        For i ← 0 to m-1 do
          Begin
              f ← 0
              For j ← 0 to L-1
                Begin
                    if ( i= = H[j])
                          f ← 1
                End
              if (f= =0)
                Begin
                    KEK ← KEK U { key at parent of subgroup i }
                     Exclude members of subgroup i from the user set S
                End
          End
Step 2:  /*  if no member is leaving from a particular position */
        For i ← 0 to m-1 do
          Begin
              f ← 0
              For j ← 0 to L-1 do
                Begin
                    if ( i= = P[j])
                          f ← 1
                End
              if (f= =0)
                Begin
                    KEK ← KEK U { key at member i }
                     Exclude ith member from all the subgroups from the user set S
                End
          End
Step 3: /* for the users at the same position in different subgroups */
        f ← 0
        For k ← 0 to L-2 do
          Begin
              if  (P[k] ≠ P[k+1])
                Begin
                    f ← 1
                    Goto Step 4
                End
          End
        For j ← 0 to m-1 do
           Begin
              if (j ≠ P[1])
                Begin
                    KEK ← KEK U { key at member j }
                    Exclude jth member from all the subgroups from the user set S
                End
           End
Step 4:    While user set S not empty do
             Begin
```

```
/*  For each user i in user set S compute new encryption key EK */
    EK ← (key at member i ) XOR (key at parent of member of i )
    KEK ← KEK U {EK}
    Exclude member i from the user set S
End
```

Protocol 2 : Computation of encryption keys for cumulative arbitrary member removal

7. Conclusion

The protocols discussed in the paper deal with two members removal(leave) and cumulative removal of arbitrary members from a secure group in an efficient manner. In our scheme server is required to store $(log_2 N * m)$ keys, along with the Group key GK, where as the scheme in [2] requires $O(N)$ keys to be stored at the server. We have shown the comparison of our scheme with the scheme proposed by Wong et al [2] with respect to computation. The binary tree concept discussed in [3] is efficiently extended to *m*-ary tree in this paper with reduced storage at user side.

References

[1] A. Bellardie, "Scalable Multicast Key Distribution", RFC 1949, May 1996.

[2] Chung Kei Wong, Mohamed Gouda, and Simon S Lam, "Secure Group Communication Using Key Graphs", Proceedings of ACMSIGCOMM, Vancouver, British Columbia, September 1998.

[3] I. Chang, R.Engel, D.Kandlur, D.Pendarakis and D.Daha. "Key management for secure internet multicast using Boolean function minimization technique". ACM SIGCOMM'99, March 1999.

[4] Debby M. Wallner, Eric J. Harder, Ryan C. Agee, "Key Management for Multicast: Issues and Architectures", Informational RFC, draft-Wallner-key-arch-ootxt, July 1997.

[5] H.Harney, C.Muckenhirn, "Group Key Management Protocol (GKMP) Architecture", RFC 2094, July 1997.

[6] H.Harney, C.Muckenhirn, "Group Key Management Protocol (GKMP) Specifications", RFC 2093, July 1997.

[7] D.McGrew and A. Sherman. "Key establishment in large dynamic groups using one way function trees".. Available at http://www.cs.umbc.edu/~sherman/papers/itse.ps, May 1998.

[8] A. Perrig, D.Song and J.Tygar, "ELK: A new protocol for efficient large-group key distribution". In Proceedings of the 2001 IEEE symposium on Security and Privacy, 2001.

[9] Ran Canetti, Benny Pinkas, "A Taxonomy of Multicast security issues", Internet Draft, May 1998.
[10] Suvo Mittra, "Iolus: A Framework for Scalable Secure Multicasting", Proceedings of ACMSIGCOMM'97, Cannes, France, pp. 277-288, 1997.

A.S.Poornima obtained her M.Tech. from VTU, Belgaum, Karnataka, India. She is presently working as an Assistant Professor in the Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur, Karnataka, India and pursuing Ph.D in the area of Cryptography and Network Security.

R.Aparna obtained her M.S. from Birla Institute of Technology, Pilani, Rajasthan, India. She is presently working as an Assistant Professor in the Department of Computer Science and Engineering, Siddaganga Institute of Technology, Tumkur, Karnataka, India and pursuing Ph.D in the area of Cryptography and Network Security.

B.B.Amberker obtained his Ph.D from Department of Computer Science and Automation, IISc., Bangalore, India. He is presently working as Professor in the Department of Computer Science and Engineering, National Institute of Technology, Warangal, AP, India.