Error Correction for a Secure Multicast Group Key Management using Gray Code

R. Varalakshmi
Ramanujan Computing Centre, Anna University
Chennai, Tamil Nadu, India

V. Rhymend Uthariaraj, PhD.
Ramanujan Computing Centre, Anna University
Chennai, Tamil Nadu, India

ABSTRACT

Key agreement protocols are designed to provide two or more specified parties communicating over public channels with a common shared secret key, which may subsequently be used to exchange information among communicating parties. Therefore, building secure key agreement protocols over open networks is essential in information security. Error Correcting Codes (ECC) is one of many tools made available for achieving data transmission. Low Density Parity-Check codes (LDPC), a linear block code which has the advantage that they provide the performance at that close to the limited capacity for many different channels and linear time complex algorithms for decoding. In this paper, we propose a new group key computation protocol that provides more security and also integrates a Gallager code and it is proved that this proposed approach takes less decoding time complexity.

General Terms

Group Key Management, Network Security.

Keywords

Key agreement protocols, Error Correcting Codes, Gallager code, Low Density Parity-Check codes, Key Computation.

1. INTRODUCTION

Many applications like pay-per–view, distribution of digital media etc., require secure multicast services in order to restrict group membership and enforce accountability of group members. A major issue associated with the deployment of secure multicast delivery services is the scalability of the key distribution scheme. This is particularly true with regard to the handling of group membership changes, such as membership departures and/or expulsions, which necessitate the distribution of a new session key to all the remaining group members. As the frequency of group membership change increases, it becomes necessary to reduce the cost of key distribution operations. One solution is to let all authorized members use a shared key to encrypt the multicast data. To provide backward and forward confidentiality (D.M. Wallner and Agee, 1999), this shared key has to be updated on every membership change and redistributed to all authorized members securely which is referred to as rekeying. The efficiency of rekeying is an important issue in secure multicast as this is the most frequently performed activity with dynamic change in the membership.

Group key must be updated with the group membership changes to prevent a new member from deciphering messages exchanged before it join the group; this is defined as backward secrecy. Group key revocation in case of one member joins or multiple members join could be achieved by sending the new group key to the old group members encrypted with the old group key. Also, group key must be must be updated with the group membership changes to prevent an old member (leaved or expelled) from deciphering current and future communication which is defined as forward secrecy. Group key revocation, when one member leaves or multiple members leave, is more complicated in case of join because of the disclosure of the old group key. The old group key is known to the leaving member(s) so there is a need to re-key the group using valid key(s) in a scalable way. The trivial scheme for rekeying a group of n members is through using individual secret key shared between the Key distribution Centre KDC and each member. This is not a simple or scalable method and consumed large bandwidth especially for large group with high membership changes: furthermore it takes more time and needs more resources per hosts than using multicasting to re-key the group.

The rest of the paper is organized as follows. Section 2 related work. The secured multicast key management model is presented in Section 3. Section 4 describes the procedure for secured key distribution. Section 5 analyses the performance of the proposed scheme. Section 6 describes the error correction using Gallager codes, followed by the conclusion in Section 7.

2. RELATED WORK

The topics of key management for multiparty communications in general networks are studied and one of efficient key tree based group key management technique called Logical Key Hierarchy (LKH) is discussed [1,2,3,4]. A new group keying method that uses one-way functions [8] to compute a tree of keys, called the One-way Function Tree (OFT). In this method, the keys are computed up the tree, from the leaves to the root. This approach reduces re-keying broadcasts to only about log n keys. The major limitation of this approach is that it consumes more space. However, time complexity is more important than space complexity.

A key update in scheme [5,11,12,13] requires $O(\log_2 N)$ messages where N is the size of the group. In this scheme each user has to store $\log_2 N$ 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 uses the LKH scheme and uses a binary tree, but with only two keys at every level. This reduces total number of keys at the server from O(N) to O(h) where h is the height of the tree. But storage at each user remains at $O(\log_2 N)$. The scheme discussed and extends the scheme to m-ary tree instead of binary tree, which reduces the user side storage from $O(\log_2 N)$ as to $O(\log_m N)$[2]. In tree based key management schemes each user shares a key called private key with the key server and key at the root of the tree is the group key which is shared by all users in the group. Other keys (other than private key and group key) are called auxiliary keys (key encryption keys) which are known only for certain subset of users and are used to encrypt new group key whenever there is a group membership change.

The scheme uses m-ary tree and at each level m keys are maintained. Whenever a node is compromised new group key is selected and distributed to other nodes. The encryption keys that are required to send new group key GK_{new} securely are computed. The new group key is distributed to group members without performing any encryptions.

Low-Density Parity-Check codes (LDPC) [9, 10] as one of many kinds, are also linear block codes that have been studied vastly in this decade. LDPC became more popular and widely developed for wider area of applications including communications and data storage. There are two different ways to represent LDPC codes; matrix representation and graphical representation. In the matrix point of view, as it is named, LDPC codes hold small number of "1" in each row and column, i.e. $W_c << n$ and $W_r << m$ for a dimension $m \times n$ parity matrix. This can provide large minimum distance of the code. However such a circumstance results a large parity check matrix. In the graph point of view, Tanner graph [7] is an efficient graphical representation of LDPC codes. There are m check nodes (cnodes; number of parity bits) and n variable nodes (v-nodes; number of bits in a codeword).LDPC codes are said to be regular if W_c is constant for every column, and $W_r = W_c$ (n/m) . If the parity matrix H is low density but the number of "1" in each row or column are not constant, the code is said to be an irregular one.

Our scheme distributes new group key to the remaining group members with minimum number of messages as compared to the scheme in [4]. In our scheme, in order to avoid the leaving members using auxiliary keys to learn the new group key, auxiliary keys are also updated.

3. SECURED MULTICAST KEY MANAGEMENT MODEL

In our scheme each member of the group is associated with a unique user ID (UID) which is a Gray code of string length n. Gray Code is a form of binary that uses a different method of incrementing from one number to the next.

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 member. Extending the scheme to m-ary tree will reduce the height of the tree reducing number of keys at each member. 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

$n = 2d = md'$

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

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 member and server, following relation has to be satisfied :

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

The Controller executes the protocol shown in Figure 2 to compute the messages using gray codes that need to be sent out after multiple members depart the group in the same round.

3.1 Representation 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 members associated with the group. Each member is assigned with Unique Identification Number (UID) which is a Gray Code of length n (where $n = \log_2 N$).

Subgroups: Each interior node containing at the maximum children nodes forms one subgroup. Subgroups at level i are assigned with keys Ki_0 to $Ki_{(m-1)}$ called Auxiliary keys at level i.

Keys: Individual member keys of any subgroup are numbered from K_0 to K_{m-1} so that all the members at position 0 of all subgroups are assigned with key K0 and members at position 1 of all subgroups are assigned with key K_1 and so on up to K_{m-1}.

KEK : Key Encryption Keys is the set, initially empty, and at the end contains the keys used to encrypt the new auxiliary keys and member keys.

$\{ G_K \} K_1$ denotes GK is encrypted with the key K_1.

|| 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:

Members u_0, u_4, u_8, u_{12} are assigned with key K_0

Members u_1, u_5, u_9, u_{13} are assigned with key K_1

Members u_2, u_6, u_{10}, u_{14} are assigned with key K_2

Members 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_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}

GK_{new} new group key

4. SECURED KEY DISTRIBUTION

The encryption keys computed using the method of [11] are used to communicate new group key to the existing nodes without actually performing any encryption. Messages send by central node to group members by using the hash of the encryption keys that are known to compromised nodes. Hence using the keys of the compromised nodes it is not possible to get any information regarding new group key. In order to avoid attackers decrypting any message in the next time interval we perform two operations. First, each remaining node along with path from the leaving point will compute new auxiliary key using the method,

F(auxiliary keys, GK_{new})$\rightarrow$ auxiliary keys XOR GK_{new}

Second, every key used to compute the hash value is incremented by one (1). In this scheme to communicate new group key securely we are not using any encryption instead all communications are by using hash values and XOR operations which will reduce the communication overhead i.e., rekeying cost is reduced.

4.1 Member Leaves

Any number of members can leave (be removed from) the multicast group from any position in the m-ary tree. The protocol shown in figure 2, depicts the computation of encryption keys for other than individual member removal.

```
Let $B_1$ = Most Significant Bit; $B_4$ = Least Significant Bit
S = Set contains remaining group members
C = 0 (Count for no. of sub group with no members leaving)

Step 1: /* No member leaving from that sub group */
    Do 1 to no. of subgroups
        If no. of same occurrence at $B_1B_2$ = No. of members in that sub group
        S = S – Group members from the sub group
        C = C + 1
        End if
    End Do
Step 2: /* No member leaving in a particular position in remaining subgroup */
    Do 1 to no. of subgroups – C
        If no. of occurrence of B3B4 = no. of subgroup – C
        S = S – Group members from the sub group
        End if
    End Do
Step 3: /* Users at same position in different sub group */
    Do 1 to no. of subgroups – C
        If no. of occurrence of $B_3B_4$ != 1
        S = S – Group members from the sub group
        End if
    End Do
Step 4: /* Users at different position in different sub group */
    Do 1 to no. of subgroups – C
        If no. of occurrence of $B_1B_2$ = 1
        S = S – Group members from the sub group
        End if
    End Do
End
```

Figure 2 : Protocol for computation of encryption keys other than individual member leave.

5. PERFORMANCE COMPARISON

To achieve message confidentiality in Secure Group Communication we require a group key and the group key should be updated whenever a node is compromised. In our scheme server is required to store ($\log_2 N$ * m) keys, along with the Group Key GK, whereas the scheme in [2] requires O (N) keys to be stored at the server. The binary tree concept discussed in [3] is efficiently extended to m-ary tree in this paper with reduced storage at user side. Each member is assigned with Unique Identification Number (UID) which is a Gray Code of length n (where n= $\log_2 N$). New Group Key is distributed to the existing nodes using hash functions and XOR operations.

5.1 SIMULATION AND RESULTS

From the simulation shown in figure 3, it is clear that the number of rekeying messages sent after member removal is reduced. The reduction is more significant when the user leaves are unpredictable where there is a reduction in the number of rekeys messages. Therefore the number of keys that has to be changed is reduced and unwanted network traffic is reduced.

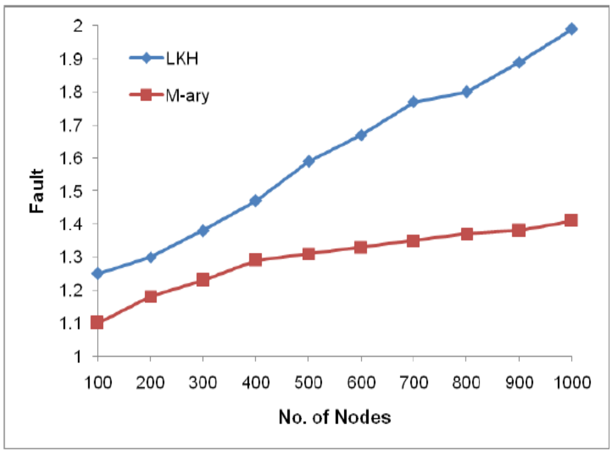

Figure 3. Reduction of No. of Rekeying Messages

6. ERROR CORRECTION USING GALLAGER CODES

Error Correcting Codes (ECC) [6] is one of many tools made available for achieving data transmission. Low Density Parity-Check codes (LDPC), a linear block code which has the advantage that they provide the performance at that close to the limited capacity for many different channels and linear time complex algorithms for decoding.

6.1 Encoder

Similar to all other linear block codes, we have the relation;

$$C_{(1\times n)}H^T_{(n\times m)} = 0 \qquad (1)$$

where C is a codeword matrix, and H is a parity check matrix. In a systematic form, C can be written as:

$$C_{(1\times n)} = \left[m_{(1\times m)} \middle| \, p_{(1\times n-m)}\right] \qquad (2)$$

Where $p_{(1\times n-m)}$ denotes the parity portion, and $m_{(1\times m)}$ denotes the message portion respectively. With

$$H = [H_1 \quad H_2] \; or \; H^T = \begin{bmatrix} H_1^T \\ H_2^T \end{bmatrix}$$ We can have;

$$CH^T = [m \mid p] \begin{bmatrix} H_1^T \\ H_2^T \end{bmatrix}$$

$$= mH_1^T + pH_2^T = 0 \qquad (3)$$

Or

$$p = mH_1^T + (H_2^T)^{-1} = 0 \qquad (4)$$

The task of the encoder is then to compute the parity matrix P that can be directly appended to the message to produce the codeword.

For the matrix H to be more manageable, the LU decomposition method can be preferably applied; i.e. [H] = [L][U]. Thus,

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & l_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} \qquad (5)$$

Let [Y] = [U][P] , then we can use forward substitution to solve [L][Y] = [M].

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} \qquad (6)$$

Finally, the backward substitution is employed to solve for P of which [U][P] = [Y] . There, we can get {pi} as needed.

$$\begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & l_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad (7)$$

6.2. Decoder

There are several methods used in decoding the LDPC codes[9,10]. Each was derived individually. These are, for instance, Believe Propagation (BP), Sum-Product (SP), and Message Passing (MP).

The Tanner graph (Fig.4) can be drawn directly from the H matrix (given in (8)) as shown be below:

$$H = \begin{Vmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{Vmatrix} \qquad (8)$$

:

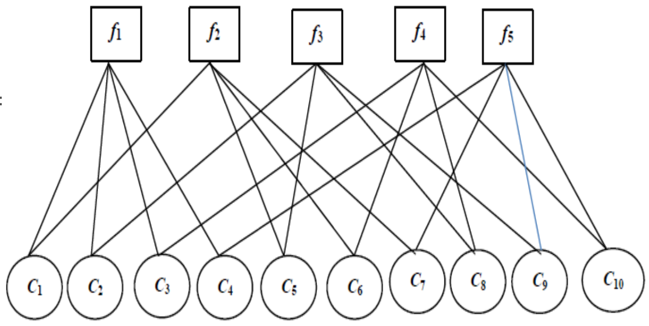

Fig. 4. Tanner graph of the H matrix given in (8)

The graph contains m check nodes (number of parity bits) and n variable notes (number of bits in a codeword). Check node fi is connected to a variable node ci if the element hij of H is a “1”.

In the Log-domain Sum-Product algorithm, the message passes between check nodes and variable nodes. In each pass the log likelihood ratio (LLR) is recorded for is probability of its likely symbol. In summary, the decoder goes through 5 steps as follows:

Step 1:

Compute the initial value of L(qij) transmitted from the variable node i to check node j; for all i; 1 ≤ i ≤ n . $L(q_{ij}) =$

$$L(C_i) = \frac{2y_i}{\sigma^2} = LLR_i = \log\left(\frac{P_{(c_i-0)|y_i}}{P_{(c_i-1)|y_i}}\right) \qquad (9)$$

Where L(ci) denotes log likelihood ratio

σ^2 denotes derivation of white noise

p(ci =x)|yi denotes probability for given input yi

Step 2:

Compute L(rji) transmitted from the check node j to variable node i; for all i; 1 ≤ i ≤ n .

$$L(r_{ji}) = \prod_{i'\in V_{j/i}} \alpha_{i'j} \emptyset(^n_1 Y) \sum_{i'\in V_{j/i}} \emptyset(\beta_{i'j}) \qquad (10)$$

Where αij= sgn{*L*(*qij*)}, and βij= *L*(*qij*).

Step 3:

Modify L(qij) and used it as the data transmitted from the variable node i to check node j; for all i; $1 \le i \le n$.

$$L(q_{ij}) = L(c_i) + \sum_{j' \in c_{i/j}} L(rj'i) \qquad (11)$$

Step 4:

Compute the soft output.

$$L(Q_i) = L(c_i) + \sum_{j \in C_i} L(r_{ji}) \qquad (12)$$

Step 5:

The soft output obtained in step 4 is then used in the hard decision as,

$\hat{c}$i = 1 if L(Qi) < 0 , otherwise cˆi = 0 .

The following proof gives the information regarding the number of changes for the different types of errors.

Lemma:

Any arbitrary LDPC codes has O(n^2) time complexity during decoding process for n bit errors.

Proof:

An error is said to occur:

1. If the values of the level 1 check nodes (i.e., key check nodes) are not zero.

2. If the computed parity bit values and received parity bit values at each level, in encoding stopping set are unequal.

We need to correct these errors. On occurrence of error in the information bits, the following procedure has to be followed. Since the number of corrupted bits and their position are unknown, we correct them step by step procedure. First we change the 1st bit of the leaf nodes from left to right. Next, we compute the new parity bit values. If the key check node values are equal to zero, then the error is corrected.

Even now, if the key check node values are unequal to zero, then the second bit of the information bit is changed and the procedure is repeated until reaching the last information bits in the leaf level. From this it is very clear that the complexity for correcting one bit error is O(n). If still error persists, the above procedure is repeated for all combination of two information bits. Now the time complexity becomes O(n+(n(n-1)/2)). Even then if the error is uncorrected, then the combination of „i‟ (i=3,4,…..,n) information bits are changed to calculate the new parity bit value, and the error is corrected. Hence the time complexity for the decoding procedure is O(n) as follows.

For example if the total number of received information bits is 4 bits and all the four information bits are corrupted, then the decoding time complexity can be computed as shown below.

$= n + (n (n – 1) /2) +((n – 1) (n – 2) / 2)+1$

$= n+((n^2 – n) /2)+(n^2 – 3n + 2) / 2)+1$

$= n^2 –n + 2$

$=O(n^2)$

7. CONCLUSION

In this paper, the scheme uses m-ary tree based group key computation protocol for n bit numbers as the key value has been proposed for creating and distributing keys in order to provide effective security in group communications. A comparison between the proposed scheme and the previous schemes were undertaken according to storage requirements at both group controller and group members and the number of updates multiple leaves. The comparison shows that the proposed scheme using the reflected code called Gray code achieves lower storage requirements and lower communication overhead at both the group controller and the group members. On the other hand, Low Density Parity-Check codes (LDPC), a linear block code which has the advantage that they provide the performance at that close to the limited capacity for many different channels and linear time complex algorithms for decoding.

8. REFERENCES

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

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

[3] Debby M. Wallner, Eric J. Harder, Ryan C. Agee, 1997, “Key Management for Multicast: Issues and Architectures”, Informational RFC, draft-Wallnerkey-arch-ootxt.

[4] H.Harney, C.Muckenhirn, 1997, “Group Key Management Protocol (GKMP) Architecture”, RFC 2094.

[5] D.McGrew and A. Sherman, 1998, “Key establishment in large dynamic groups using one way function trees”.

[6] C Berrou, A. Glavieux and P. Thitimajshima, 1993, “Near Shannon limit error-correcting Coding and Decoding,” Proc. IEEE Int. Conf.Comm., pp.1064-1070.

[7] R.M. Tanner, 1981, “A Recursive Approach to Low Complexity Code,” IEEE Trans. Information Theory, pp.533-547.

[8] J.L. Fan, 2000, “Array Codes as low-density parity-check codes,” Proc. 2nd Int. Symp. Turbo Code, Beit, France, pp. 543-546.

[9] R. Gallager, 1962, "Low-density Parity-check Code," IRE Trans. Information Theory, pp.21-28.

[10] Jin Lu., and Jose M.F.Moura, 2010, "Linear Time Encoding of LDPC Codes" IEEE Trans. Inf. Theory, vol. 57, no. 1, pp. 233-249.

[11] Ran Canetti, Benny Pinkas, 1998, "A Taxonomy of Multicast security issues", Internet Draft.

[12] SuvoMittra, 1997, "Iolus: A Framework for Scalable Secure Multicasting", Proceedings of ACMSIGCOMM'97, Cannes, France, pp. 277-288.

[13] Rafaeli and D. Hutchinson, 2003, "A Survey of Key Management for Secure group communication", ACM Computing Surveys, 35:309-329.

[14] D. J. C. Mackay, 1999, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inform. Theory, vol. 45, no. 2, pp. 399–431.

[15] G. Berrou and A. Glavieux and P. Thitimajshima, 1993, "Near Shannon limit error-correcting coding: Turbo codes," Proc. 1993, International Conf. Comm, pp. 1064-1070, Geneva, Switzerland.

[16] J. Lu, Jos´e M. F. Moura, and H. Zhang, 2003, "Efficient encoding of cycle codes: a graphical approach," Proc. of 37th Asilomar Conference on Signals, Systems, and Computers, pp. 69-73, PACIFIC GROVE, CA, Nov. 9-12.

[17] J. Katz and M. Yung, 2007, "Scalable Protocols for Authenticated Group Key Exchange," J. Cryptology, vol. 20, pp. 85-113.

[18] Wei Yu, Yan (Lindsay) Sun, Member, IEEE, and K.J. Ray Liu, Fellow, IEEE, 2007, "Optimizing the Rekeying Cost for Contributory Group Key Agreement Schemes", IEEE transactions on dependable and secure computing, vol. 4, no. 3.

[19] S. Mittra, 1997, "Iolus: A framework for scalable secure multicasting," Proceedings of the ACM SIGCOMM, vol. 27, no. 4, pp. 277-288, New York.

[20] R. Varalakshmi, Dr. V. Rhymend Uthariaraj, 2011, "A New Secure Multicast Group Key Management Using Gray Code" IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011, pp 85-90.