Journal of Theoretical and Applied Information Technology
10th August 2013. Vol. 54 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGN OF GROUP HIERARCHY SYSTEM FOR MULTICAST COMMUNICATION

[1]R. VARALAKSHMI, [2]Dr. V. RHYMEND UTHARIARAJ
[1]Ramanujan Computing Centre, Anna University, Chennai, India
[2]Ramanujan Computing Centre, Anna University, Chennai, India
E-mail: [1]rvaralakshmi697@gmail.com, [2]rhymend@annauniv.edu

ABSTRACT

Secure and reliable group communication is an active area of research. Its popularity is fuelled by the growing importance of group-oriented and collaborative applications. The central research challenge is secure and efficient group key management. The present paper is based on the Design of Group hierarchy system for multicast communication using the most popular absolute encoder output type code named Gray Code. The focus is of two folds. The first fold deals with the reduction of computation complexity which is achieved in our protocol by performing fewer multiplication operations during the key updating process. To optimize the number of multiplication operations, the Fast Fourier Transform (FFT), divide and conquer approach for multiplication of polynomial representation of integers, is used in this proposed work. The second fold aims at reducing the amount of information stored in the Group Center and group members while performing the update operation in the key content. Comparative analysis to illustrate the performance of various key distribution protocols is shown in this paper and it has been observed that this proposed algorithm reduces the computation and storage complexity significantly.

Keywords: *Multicast Group Key Management, Group Hierarchy System, Fast Fourier Transform, Polynomial Integer Multiplications, Computation Complexity, Storage Complexity.*

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 [2]. 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 (departed or expelled) from deciphering current and future communication which is defined as forward secrecy[2]. Group key revocation, when one member departs or multiple members depart, 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

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

more time and needs more resources per hosts than using multicasting to re-key the group.

In this paper we study the problem of key management for multimedia multicast services. We begin in Section2 by introducing the concepts of key management in multicast communication and present a gray code[1] based group key management scheme that will be used later in the paper in Section 3. In Section 4, we introduce a HCF based key distribution protocol for multicast key management to minimize the computation overhead, storage complexity and the number of rekeying operations. Section 5 analyses the performance of the proposed scheme, followed by the conclusion in Section 6.

2. KEY MANAGEMENT IN MULTICAST COMMUNICATION

During the design of a multicast application, there are several issues that should be kept in consideration when choosing a key distribution scheme.

The problem of designing efficient key updating schemes [1-10] has seen recent attention in the literature. One approach for achieving scalability is to apply hierarchical subgroups and map the KEKs to a logical hierarchy. The hierarchy-based approach to group rekeying was originally presented by Wallner et al. [3], and in-dependently by Wong et al. [18]. Due to the hierarchy structure, the communication overhead is O(log n), while the storage for the center is O(n) and for the receiver is O(log n). We note that the O notation is presented to indicate that the constant factors are implementation dependent.

In most of the existing Key Management schemes, different types of group users obtain a new distributed multicast Group key which is used for encrypting and decrypting multimedia data for every session update. Among the various works on key distribution, Maximum Distance Separable (MDS) [12] method instead of using encryption algorithms focuses on error control coding techniques for distributing re-keying information. In MDS, the key is obtained based on the use of Erasure decoding functions [13] to compute session keys by the GC/group members. Moreover, the Group center generates n message symbols by sending the code words into an Erasure decoding function. Out of the n message symbols, the first message symbol is considered as a session key and the group members are not provided with this particular key alone by the GC. Group members are given the (n − 1) message symbols and they compute a code word for each of them. Each of the group members uses this code word and the remaining (n − 1) message symbols to compute the session key. The main limitation of this scheme is that it increases both computation and storage complexity. The computational complexity is obtained by formulating lr + (n − 1)m where lr is the size of r bit random number used in the scheme and m is the number of message symbols to be sent from the group center to group members. If lr = m = l, computation complexity is nl. The storage complexity is given by (log2 L + t) bits for each member. L is number of levels of the Key tree. Hence Group Center has to store n (log2 L + t) bits.

Secure communication using the extended Euclidean algorithm [14] was proposed for centralized secure multicast environments. The main advantage of this algorithm is that only one message is generated per rekeying operation and only one key is stored in each user's memory. In this algorithm, two values (δ, L) are computed in the intermediate steps of GC. The main limitation of the Euclidean algorithm is that the two computed values must be relatively prime. If this is not the case, then the algorithm fails in which the user cannot recover the secret information sent by GC. Also, the time taken for defining a new multiplicative group is high, whenever a new member joins or depart the multicast operation. This approach is only suitable for a star based key management scheme.

The Data Embedding Scheme proposed in [15] is used to transmit a rekeying message by embedding the rekeying information in multimedia data. In this scheme, the computation complexity is O(log n). The storage complexity also increases to the value of O(n) for the server machine and O(log n) for group members. This technique is used to update and maintain keys in a secure multimedia multicast via a media dependent channel. One of the limitations of this scheme is that a new key called an embedding key has to be provided to the group members in addition to the original keys, which causes a lot of overheads. A level homogeneous key tree [16] based key management scheme was proposed in [17] to reduce computation and storage complexity. A Key management scheme using key graphs has been proposed by Wong Gouda [18] which consists of the creation of secure group and basic key management graphs scheme using a Star and Tree based method. The limitation of this approach is that scalability is not achieved. A new group keying method that uses one-way functions [19] to compute a tree of keys, called the One-way Function Tree (OFT) algorithm has been proposed

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

by David and Alan. In this method, the keys are computed up the tree, from the departs 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, the time complexity is more important than space complexity. The storage complexity of GC is 2nK and group member is LK , where K is the key size in bits. In our work, we focused on reduction of computation of both time complexity as well as storage complexity.

Wade Trappe and Jie Song proposed a Parametric One Way Function (POWF) [20] based binary tree Key Management. Each node in the tree is assigned a Key Encrypting Key (KEK) and each user is assigned to a leaf and given the IKs of the nodes from the leaf to the root node in addition to the session key. These keys must be updated and distributed using top down or bottom up approach. The storage complexity is given by (logτn)+2 keys for a group center. The amount of storage needed bythe individual user is given as (τL+1−1) / τ−1 keys. Computation time is represented in terms of amount of multiplication required. The amount of multiplication needed to update the KEKs using the bottom up approach is τ logτ n−1. Multiplication needed to update the KEKs using the top down approach is (τ−1) logτ n(logτ n+1) / 2 . This complexity can be reduced substantially if the numbers of multiplications are reduced. In P. Vijayakumar et al.,[21] proposed Centralized key distribution protocol using greatest common divisor method using a cluster based Key Management Scheme that reduces computation time by reducing the number of multiplications required in the existing approaches. They use the Karatsuba fast multiplication algorithm to optimize the multiplication operations used in the key distribution protocol in the GC.

Therefore, in this paper we propose a new Group hierarchy based Key Management Scheme using Gray Code [1] that reduces computation time as well as storage by reducing the number of multiplications required in the existing approaches, and also refreshes the session key for secure communication. We also use the fast fourier transform polynomial integer multiplication algorithm which yields better results than the previous scheme to optimize the multiplication operations used in the key distribution protocol in the GC. The proposed method also reduces the amount of information that needs to be stored for updating the keys when there is a change in the group membership. Our proposed algorithm is suitable for single or batch join/depart operation which is achieved by refreshing the session key.

3. DESIGN OF GROUP HIERARCHY BASED MULTICAST COMMUNICATION

Scalability can be achieved in this proposed key distribution approach by applying this scheme in a Group hierarchy based key management scheme to update the GK and SGK. Fig. 1 shows a Group hierarchy in which the root is the Group key (kg), leaf nodes are individual keys, and the intermediate level is Sub Group Key (SGK-k_ε). The hierarchy shown in Fig. 1 consists of only three levels. The lowest level (0th level) is the group key. The next higher level (1st level) contains the shared secret keys, $k_\varepsilon i$, where εi = 1, 2, . . . , n. The last level (2nd level) is the user's level, where M number of users are grouped into k Groups, Hk. To issue new keys upon a user event, the main task is to identify the keys that need to be changed. So, 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.

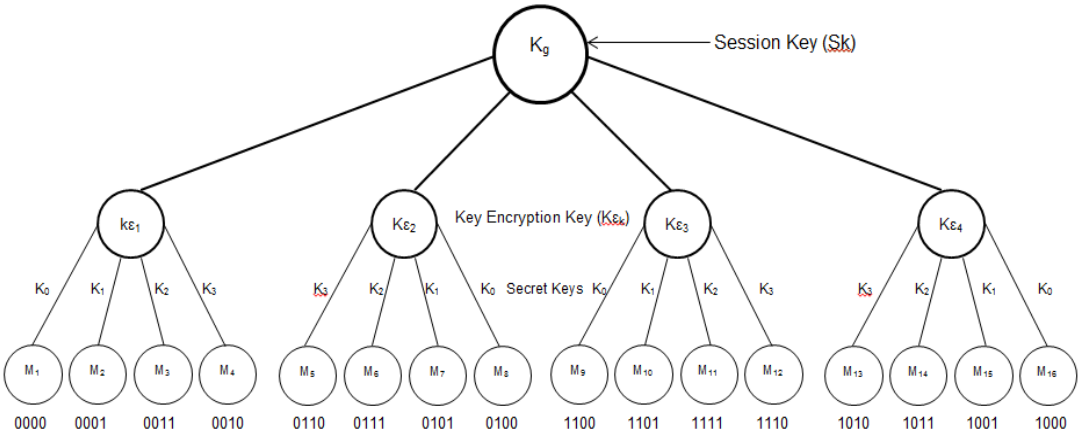

Figure 1. Group Hierarchy Based Key Management Scheme Using Gray Code

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

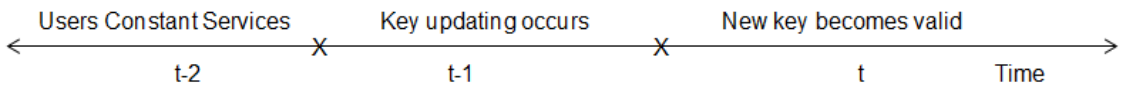

Figure 2. Time Intervals

Table 1. Decimal, Natural Binary And Gray Code Representation

Decimal	Natural Binary	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Table 1 depicts the Decimal, Natural Binary and Gray Code Representation. With Gray Code, only one bit changes state from one position to another. The length of the UID depends upon the size of the multicast group. Each Groups is attached to the upper level (1st level) node and in turn with the group key node. When the number of joining users exceeds the Groups size, a new node is created from the root to form the second Groups . The number of Groups s formed is based on the Groups size M which is fixed by GC and the number of joining users. If the Group hierarchy based key management consists of N number of users M1, M2, . . . , MN and each Groups size is of size M then there will be ⌈ N/M⌉ Groups s. In this Group hierarchy based key management scheme, updating is necessary for each rekeying operation used for member depart and member join operations.

Due to the dynamic nature of the group, and the possible expiration of keying material, it is necessary to update both the SK and KEKs using rekeying messages. The Four operations involved are Key Refreshing, GC Initialization, key updating when a new user joins the service, and key updating when a user departs the service. In the discussions that follow, we use an integer-valued time index to denote the time intervals during which fundamental operations occur, and assume that there is a system-level mechanism that flags or synchronizes the users to the same time frame. We shall always use the time index to denote the interval for which the new key information will become valid. Time interval will correspond to the time interval during which the new key information is being transmitted. Further, time interval corresponds to the interval of time during which a new member contacts the service provider wishing to join, or a current member announces to the service provider his desire to depart the service. We have depicted these cases in Fig. 2. Observe that it is not necessary that the time intervals have the same duration.

3.1 Key Refreshing

Refreshing the session key is important in secure communication. As a session key is used, more information is released to an adversary, which increases the chance that a SK will be compromised. Therefore, periodic renewal of the session key is required in order to maintain a desired level of content protection. By renewing keying material in a secure manner, the effects of a session key compromise may be localized to a short period of data.

The crypto-period associated with a session key is governed by many application-specific considerations. First, the value of the data should be examined and the allowable amount of unprotected (compromised) data should be addressed.

Since the amount of data encrypted using KEKs is usually much smaller than the amount of data encrypted by a session key, it is not necessary to refresh KEKs. Therefore, KEKs from the previous time interval (t-1) carry over to the next time interval. In order to update the session key $k_g(t-1)$ to a new session key kg(t), the group center generates kg(t) and encrypts it using the KEK $k_\varepsilon(t)$. This produces a rekeying message $\alpha g(t) = E(k_\varepsilon(t), kg(t))$, where we use E(k,m) to denote the encryption of m using the key K. The message αg(t)is sent to the users.

3.2 GC Initialization

Initially, the GC selects a large prime number p and q, where p > q and q ≤ p/4. The value p helps in defining a multiplicative group Z_p^* and q is used to fix a threshold value δ, where δ = a + q. The value a is a random element from the group $Z_p^*$and hence

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

when the 'a' value increases, the value of δ also increases.

3.3 Member Join

In multimedia services, such as pay-per-view and video conferences, the group membership will be dynamic. Members may want to join and depart the service. It is important to be able to add new members to any group in a manner that does not allow new members to have access to previous data. In a pay-per-view system, this amounts to ensuring that members can only watch what they pay for, while in a corporate video conference there might be sensitive material that is not appropriate for new members to know.

3.3.1 Key Updating process steps at Group Center Side

1 Initially, GC selects a random element g from z_p^*

2 GC now computes the shared secret key $k_\varepsilon = k_g^a \mod p$.

3 The GC calculates $L=\prod_{i=1}^{m} k_i^m$

4 The GC computes a HCF value of (δ, L) by using the extended Euclidian algorithm described in [27] from which it finds x, y, b such that $x\times \delta + y \times L = b$.

5 The GC multicasts k_g, x, p, q and d to the group members.

3.3.2 Key Updating process steps at User Side

1 Computes x1 using the relation $x \mod ki = x1$.

2 Computes δ using $x1^{-1} \mod k_i = \delta$.

3 Performs the following operation to find the shared secret key.

$k_g^{b\times\delta} / k_g^q \mod p = k_g^{(b\times\delta)-q} \mod p = k_\varepsilon$.

The k_ε obtained in this way must be equal to the k_ε computed in Step 2 of GC.

Upon receiving all the above information (k_g, u, p, q, b) from the GC, an authorized user u_i of the current group executes the following steps to obtain the new group key k_g .

Suppose that, during time t-2 interval, a new user contacts the service desiring to become a group member. If there were n-1 users at time t-2 then there will be n users at time t.

During time interval t-1, the rekeying information must be distributed to the n-1 current members. Observe that we must renew both the SK and KEK in order to prevent the new user for accessing previous rekeying messages and to prevent access to prior content.

3.4 Member Departure

The protocol shown in figure 3, depicts the computation of encryption keys for member departure using gray code.

```
Let B1 = Most Significant Bit; B4 = 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 B1B2 = 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 B3B4 != 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 B1B2 = 1
S = S – Group members from the sub group
End if
End Do
End
```

Fig.3.: Protocol For Computation Of Encryption Keys For Member Departure Using Gray Code.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Let us consider the case when user un departs the group at timeframe t-1. Since user un knows kg(t-1) and kε(t-1) these keys must be renewed. First kε is renewed. Next, the session key kg(t) is updated. The GC forms a new SKkg(t) and encrypts using the new KEKkε(t) to form αg(t)= E(kε(t), kg(t)). This message is then sent to the users.

For example, if a member M7 with user ID (0101) in Groups 2 from the Fig. 1 departs the group, then the keys on the path from his leaf node to the hierarchy's root node must be changed.

Hence, only the keys k_ε and kg will become invalid. Therefore, these two keys must be updated. In order to update these two keys, two approaches are used in the member's departure operation. In the first approach, updating of the sub group key, kε2for the Groups 2 is performed as given in Algorithm 1. When a member M7user ID (0101) departs from the service, GC computes the protocol given in Fig.3, step 3 Users at same position in different sub group L(k5,8) for the existing users using their own secret keys which are kept in GC. When computing L(k5,8) for the members M5 user ID (0110), M6 user ID (0111), and M8user ID (0100) the GC uses k3, k2, and k0 which are the secret keys for the remaining members of all Groups s . Since the secret key k1 is known to the member M7with user ID (0101) who had left from the service. GC is not using the secret key k1 when it computes the function L(k5,8) for the members M5 user ID (0110), M6 user ID (0111), and M8 user ID (0100).

However, the computation time of $L(k_{5,8})$ can be reduced by dividing the k_ε by $k_1$as shown in Step 1 of Algorithm 1 rather than multiplying all users secret key once again. Next, the GC computes δ, HCF value of (δ, $L(k_{5,8})$) and generates a multicast message as indicated in Step 4 of Algorithm 1 and sends the message to all the existing members of the Groups in order to update the new SGK k_ε`.

Algorithm 1.

1 $L(k_{5,8}) = L(k_{5,8}) / k_1$

2 GC computers the new kg, q and computers δ and kε` values.

3 Now GC computes HCF value of (δ, $L(k_{5,8})$) and finds out x, y, b values.

4 Finally GC multicasts kg, x, p, q, and b to the existing group members. Group members M5 user ID (0110), M6 user ID (0111), and M8 user ID (0100) executes the following steps to obtain the new sub group key kε2.

5 Compute $x \mod k_i = x1$.

6 Compute $x1^{-1} \mod k_i = \mu$.

7 Perform the following operation to find the shared secret key.

8 $k_g^{b\times\delta} / k_g^{q} \mod p = k_g^{(b\times\delta)-q} \mod p = k_{\varepsilon 2}$.

After updating the above SGK successfully, GC has to use the second approach in order to update the group key kg using a different procedure as explained below. The new group key kg is used to encrypt the multimedia data. For updating the GK, GC generates a new group key from Z_p^*, with a condition that the new group key $k_g1<k_\varepsilon i$. If this condition is not satisfied then append a value 1 in front [1] of $k_\varepsilon i$ in order to make $k_\varepsilon i$ a greater value than kg1. Every time a new Groups is created its corresponding SGK is multiplied with all other SGKs and the result is stored in a temporary variable X. Therefore whenever a new Groups is created, only the new $k_\varepsilon i$ of the newly created Groups is multiplied with the value X which is stored in GC. Hence only one multiplication is needed for updating the GK. Similarly when an existing Groups is completely deleted X is divided by the corresponding ε i value and hence only one division is necessary for updating the GK. In order to understand the key updation when a single member departs a group, consider an example using Fig. 1 where only one member M7with user ID (0101) is allowed to depart the Groups (Groups 2). In this case, ε2 must be updated and let the updated ε2 be represented as ε21. In order to update ε2, the GC must divide X by ε2 first and then the result must be multiplied with the newly computed ε21 and the final result is stored in the variable X. This X is added with the newly generated group key kg1 to obtain k_ε and the rekeying message is formed by using the equation $k_\varepsilon = k_g1 + X$. In this way, member depart operations are handled effectively by reducing the number of multiplications/divisions.

The resultant value k_ε is broadcast to the remaining members of the group. The members of the group can recover the updated group key with the help of εi using the relation,

$k_\varepsilon \mod (k_\varepsilon i) = k_g1$.

The key strength of our algorithm is that the scalability increases sufficiently. The number of keys to be used by the GC and group members are reduced in comparison to the other existing approaches [1, 2, 15, 4, 22, 6]. Each user has to store 3 keys, since the hierarchy described in the proposed algorithm has 3 levels. If the numbers of Groups s are c and each Groups consists of cn users, then the storage complexity of GC is (cn × c) +

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

tc + 1, where tc is used to denote the total number of L (ki,j) and $k_{\varepsilon}i$ used for every Groups that are stored in GC and 1 represents group key storage area.

4. SECURITY ANALYSIS

To ensure multicast or broadcast security, group key management should satisfy four security properties

1. Non-group confidentiality,
2. Collusion freedom,
3. Future confidentiality (forward secrecy), and
4. Past confidentiality (back secrecy).

This Group hierarchy (proposed scheme) satisfies the above said properties.

5. PERFORMANCE EVALUATION

Assumption Let L=$\prod_{i=1}^{m} k_i^m$ be a multiplication function which is used for member join operation, where ki = secret is the key of a user. Now, 'σi' is the size of the k_i, where i = 1, 2, 3, . . . , n (n = size of the group).

Algorithm. Consider a scenario, where σ1 = 2 and σ2 = 2. A set of 'σi' are multiplied to form L as shown in Step 3 in Section 3.3 using the traditional multiplication operation present in most of the existing key distribution approaches [4, 13]. In order to multiply σ1 and σ2 using traditional multiplication operation, the algorithm takes 4 multiplication operations. In general, when two 'σ ' digit numbers are to be multiplied, it takes (σ2) multiplication operations in order to obtain the solution.

For optimizing the number of multiplication operations used for computing there exist faster multiplication algorithms, based on the fast Fourier transform, a divide and conquer approach [11,24–26] is used in our proposed key distribution algorithm. The idea is : multiplying two numbers represented as digit strings is virtually the same as computing the convolution of those two digit strings. Instead of computing a convolution, one can instead first compute the discrete Fourier transforms, multiply them entry by entry, and then compute the inverse Fourier transform of the result. Based on this idea, the number of multiplication operations to be performed in total to obtain the solution for the 'σ ' digit number will be O σ (n log (n)).

Therefore it is faster than the traditional multiplication, which requires σ2 single-digit products and the complexity is O σ log24 . The fast Fourier transform multiplication approach works well when the value of σ > 4000 digits. However, if the number of digits of σ < 16, this algorithm shall not show a significant difference. In order to optimize the use of the fast Fourier transform multiplication approach, the group size in our proposed key distribution algorithm can have 16-digits, 32-digits, 64-digits, 128-digits, etc. In the proposed algorithm given in Section 3.4, we have analyzed and tested the algorithm for a group size p as 16-digit, 32-digit and 64-digit prime numbers. The key values used in our algorithm are 16 and 32 digit numbers.

Theorem 1. The number of multiplications in the computation of L is in the order of O(ω log (ω)) when fast fourier transform divide and conquer multiplication is employed for the key computation process where the key size is a n digit number.

Proof :Two integers A and B, of length n represented as a polynomial in base x. It is important to stress at this stage that the length n has to be a power of two (n is even). In the implementation there will be some processing to ensure that n is always even.

A(x) = A0+ A1x + A2x2 + ... + An-1 xn-1

B(x) = B0+ B1x + B2x2 + ... + Bn-1 xn-1

Split A, and B in the following manner:

A0= a0+ a2 + ... + an-2 and A1 = a1 + a3 + ... + an-1

B0= b0 + b2 + ... + bn-2 and B1 = b1 + b3+ .....+ bn-1

Both halves are equal to ω/2 as the length n is a power of two. However, one half possesses the even positions and the other the odd positions. Given a sequence A = (a0, a2, … , aω-2), compute its Fourier transform according to the formulae

Choosing for ωk the complex roots of unity

(Note: i is equal to -1)

$$W_k = exp\left(\frac{2ik\pi}{2n}\right) = \omega^k, \qquad \omega = exp\left(\frac{2i\pi}{2n}\right)$$

This formulae is used to compute the complex roots of unity needed to evaluate thenumber at 2n distinct points.

This would give us:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$F_{2n}(A) = (c_0, c_1, \ldots, c_{2n-1}), \quad b_k = \sum_{j=0}^{2n-1} a_j . \omega$$

From above we get the sequence F2n(B) = (d0, d1, …, d2n-1),

The two sequences are multiplied together and produce the third sequence E

E = (e0, e1, …, e2n-1), where ek = ak * bk

On sequence E, we use the inverse Fourier Transform. This produces the sequence G

$$G = (g_0, g_1, \ldots, g_{2n-1}), \quad g_k = \sum_{j=0}^{2n-1} e_j . \omega^{-jk}$$

Finally, dividing each of the resulting integers by 2n will give us the coefficients that construct the product of the multiplication.

Time Complexity of FFT = T(ω)

T(ω) = 2T(ω/2) +O(ω) = O(ωlog ω).

In order to determine whether the Fourier algorithm performs close to its theoretical expectation or not, the Fourier algorithm is run 10,000 times on multiplicands of length of up to the eighth power of two (256) and the results are shown in Table3. Then, the average time for each power of two is divided by n log n (where n is the length of the multiplicands). If this ratio (time / n log n) gets smaller and smaller when increasing the length of the multiplicands then the Fourier implantation is said to be behaving close to our theoretical efficiency.

Table 1.Performance of FFT

Integer Length	Operation Time (ms) FFT	n log n	Big O ratio or (time / n log n)
2	0.007	2	0.003500
4	0.013	8	0.001625
8	0.024	24	0.001000
16	0.047	64	0.000734
32	0.092	160	0.000575
64	0.191	384	0.000497
128	0.398	896	0.000444
256	0.841	2048	0.000410

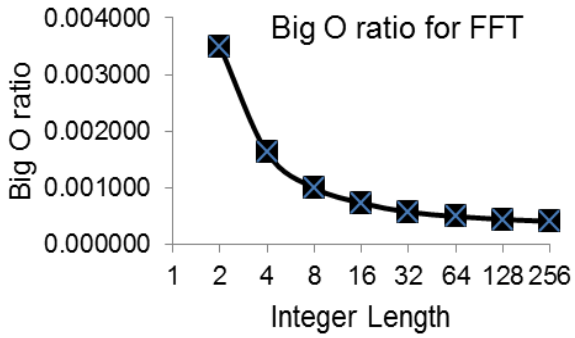

Fig.4.Big O ratio for FFT

These results in Figure 4 show that the 'Big O ratio' starts of at 0.00350 and gradually decreases to 0.000410 at integers of length 256. It is predicted to keep decreasing for greater powers of two than 256. It will eventually reach a constant c0. Therefore, the implementation of this FFT algorithm for our scenario in order to reduce the number of multiplications is considered successful as it is so close to our theoretical expectation. Most of the applications like PAY-TV, Video-on-demand, sporting events are fully based on the idea of reducing the computation time where our proposed protocol will be more suitable.

6. CONCLUSION

A comparison between the proposed scheme and the previous schemes were undertaken while providing secure multimedia multicast through effective key management techniques. For this purpose a new Group hierarchy-based key distribution protocol (key value = 'n' bit numbers) has been proposed in this paper. The proposed algorithm has two dimensional focuses—minimal computation complexity and minimal storage complexity. The comparison shows that the proposed scheme using the reflected code called Gray code [1] achieves lower storage requirements at both the group controller and the group members.

Gray Code is the most popular Absolute encoder output type because its use prevents certain data errors which can occur with Natural Binary during state changes. Therefore Gray Code is more secure than binary in encoder applications. On the other hand, with regard to the storage complexity, the amounts of keys stored by GC and group members are reduced substantially by employing the Group hierarchy approach. Further extensions to this work are to devise techniques for reducing the communication complexity which is the number of keys to be sent from GC to the group members' area in order to recover the updated keying information and to reduce rekeying cost for batch join and batch

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

depart operations. The conceptual idea has been discussed in this paper. Implementation is being carried out.

REFERENCES

[1] R. Varalakshmi, Dr.V.RhymendUthariaraj, "A New Secure Multicast Group Key Management Using Gray Code", Paper No: 978-1-4577-0590-8/11, IEEE-International Conference on Recent Trends in Information Technology, ICRTIT 2011,MIT, Anna University, Chennai. June 3-5, 2011.

[2] 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.

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

[4] Mingyan Li, R. Poovendran, A. David McGrew, Minimizing center key storage in hybrid one-way function based group key management with communication constraints, Information Processing Letters (2004) 191–198. Elsevier.

[5] R. Poovendran, J.S. Baras, An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes, IEEE Transactions on Information Theory 47 (2001) 2824–2834.

[6] Sandeep S. Kulkarni, BezawadaBruhadeshwar, Key-update distribution in secure group communication, Computer Communications 33 (6) (2010) 689–705. Elsevier.

[7] BezawadaBruhadeshwar, Kishore Kothapalli, MaddiSreeDeepya, Reducing the cost of session key establishment, ARES (2009) 369–373.

[8] BezawadaBruhadeshwar, Kishore Kothapalli, M. Poornima, M. Divya, Routing protocol security using symmetric key based techniques, ARES (2009) 193–200.

[9] Shiuh-Jeng Wang, Yuh-Ren Tsai, Chien-Chih Shen, Pin-You Chen, Hierarchical key derivation scheme for group-oriented communication systems, International Journal of Information Technology, Communications and Convergence 1 (1) (2010) 66–76.

[10] Mohsen Imani, Mahdi Taheri, M. Naderi, Security enhanced routing protocol for ad hoc networks, Journal of Convergence 1 (1) (2010) 43–48.

[11] J. M. Pollard. The Fast Fourier Transform in a Finite Field. Mathematics of Computation, 25(114):365-374, April 1971.

[12] Mario Blaum, JehoshuaBruck, Alexander Vardy, MDS Array codes with independent parity symbols, IEEE Transactions on Information Theory 42 (2) (1996) 529–542.

[13] LihaoXu, Cheng Huang, Computation-efficient multicast key distribution, IEEE Transactions on Parallel and Distributed Systems 19 (5) (2008) 1–10.

[14] J.A.M. Naranjo, J.A. Lopez-Ramos2, L.G. Casado, Applications of the extended Euclidean algorithm to privacy and secure communications, in: Proceedings of the 10th International Conference on Computational and Mathematical Methods in Science and Engineering, CMMSE, 2010.

[15] Wade Trappe, Jie Song, RadhaPoovendran, K.J. Ray Liu, Key distribution for secure multimedia multicasts via data embedding, in: IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, 2001, pp. 1449–1452.

[16] J.S. Lee, J.H. Son, Y.H. Park, S.W. Seo, Optimal level-homogeneous tree structure for logical key hierarchy, in: Proc. of IEEE Conference on Communication System Software and Middleware Workshop, COMSWARE, 2008.

[17] Dong-Hyun Je, Jun-Sik Lee, Yongsuk Park, Seung-Woo Seo, Computation-and-storage-efficient key tree management protocol for secure multicast communications, Computer Communications 33 (6) (2010) 136–148.
Elsevier.

[18] C. Wong, M. Gouda, S. Lam, Secure group communications using key graphs, IEEE/ACM Transactions on Networking 8 (2002) 16–30.

[19] A. David McGrew, Alan T. Sherman, Key establishment in large dynamic groups using one-way function trees, IEEE Transactions on Software Engineering 29 (5) (2003) 444–458.

[20] Wade Trappe, Jie Song, RadhaPoovendran, K.J. Ray Liu, Key management and distribution for secure multimedia multicast, IEEE Transactions on Multimedia 5 (4) (2003) 544–557.

[21] P. Vijayakumar et al,Centralized key distribution protocol using the greatest common divisor method, Computers and Mathematics with Applications (2012).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[22] Ng W. Hock Desmond, M. Howarth, Z. Sun, H. Cruickshank, Dynamic balanced key tree management for secure multicast communications, IEEE Transactions on Computers 56 (5) (2007) 590–605.

[23] Wade Trappe, Lawrence C. Washington, Introduction to Cryptography with Coding Theory, second ed., Pearson Education, 2007, pp. 66–70.

[24] M. Scott. An Implementation of the Fast-Fourier Multiplication Algorithm. Technical Rep ort CA-0790,Dublin City University, 1990.

[25] Tom St Denis, BigNum Math Implementing Cryptographic Multiple Precision Arithmetic, SYNGRESS Publishing, 2003.

[26] Richard Cz. Singleton, On computing the fast fourier transform, Communications of the AGM 10 (1967), 647-654.