Computation and Communication Efficient Key Distribution Protocol for Secure Multicast Communication

P.Vijayakumar[1], S.Bose[2], A.Kannan[3] and L.Jegatha Deborah[1]
[1]P.Vijayakumar, Department of Computer Science and Engineering, University College of Engineering Tindivanam, Tindivanam, India-604 001. [e-mail: vijibond2000@gmail.com]
[2]S.Bose, Department of Computer Science and Engineering, Anna University Chennai, Chennai, India- 600 025. [e-mail: sbs@cs.annauniv.edu.]
[3]A.Kannan, Department of Information Science and Technology, Anna University Chennai, India- 600 025. [e-mail: kannan@annauniv.edu.]

*Corresponding author: P.Vijayakumar

Received December 1, 2012; revised February 21, 2013; accepted March 21, 2013; published April 30, 2013

Abstract

Secure multimedia multicast applications involve group communications where group membership requires secured dynamic key generation and updating operations. Such operations usually consume high computation time and therefore designing a key distribution protocol with reduced computation time is necessary for multicast applications. In this paper, we propose a new key distribution protocol that focuses on two aspects. The first one aims at the reduction of computation complexity by performing lesser numbers of multiplication operations using a ternary-tree approach during key updating. Moreover, it aims to optimize the number of multiplication operations by using the existing Karatsuba divide and conquer approach for fast multiplication. The second aspect aims at reducing the amount of information communicated to the group members during the update operations in the key content. The proposed algorithm has been evaluated based on computation and communication complexity and a comparative performance analysis of various key distribution protocols is provided. Moreover, it has been observed that the proposed algorithm reduces the computation and communication time significantly.

Keywords: Multicast Communication, Key Distribution, One Way Hash Function, Computation Complexity, Karatsuba Multiplication.

http://dx.doi.org/10.3837/tiis.2013.04.016

1. Introduction

Multimedia services such as pay-per-view, video conferences, sporting event, audio and video broadcasting are based upon multicast group communication where multimedia messages are sent to a group of members in a secured manner. In such a scenario, groups can be classified into static and dynamic groups. In static groups, membership of the group is predetermined and does not change during the communication. On the contrary, in dynamic groups, membership can change during multicast communication. Therefore, in a dynamic group communication members can join or depart from the services during any time intervals. When a new member joins into the service, it is the responsibility of the Group Centre (GC) to prevent new members from having access to the previous data and thus providing backward secrecy in a secure multimedia communication. Similarly, when an existing group member leaves from any group, he/she should not have further access to data and this is called forward secrecy. In this model, the GC also takes care of the responsibility of distributing the Secret key and Group key to the group members.

The first aspect of the algorithm proposed in this paper focuses on dynamic groups where the group membership often changes and hence dynamic key generation and updating operations are performed. In reality, such operations consume large computation time and hence an effective key distribution protocol has to be designed in multimedia applications. The larger computation complexity in such operations is achieved by using modular exponentiation, GCD, ordinary multiplication and division operations in a binary tree, star and cluster tree based key distribution schemes. To facilitate the reduction in computational complexity, this paper proposes a new ternary tree key hierarchy which performs less numbers of multiplication operations. Moreover, reducing the computation complexity can also be performed by improving the speed of multiplication operations used in the key updating operations. Since, optimizing the number of multiplication operations is a critical performance improvement, this is also addressed by the proposed algorithm. The proposed algorithm utilizes the existing Karatsuba based fast multiplication algorithm for effective multiplication since it reduces the computation time compared to the existing algorithms.

In multimedia applications, the group communication is facilitated through group key distributions and such keys have to be updated often for providing forward and backward secrecy. Such group key interchange (GC and group members) must also reduce the communication complexity. This communication complexity is normally estimated by the number of data bits that are transmitted between the GC and the group members. The second aspect of the proposed algorithm aims at reducing the communication complexity which is facilitated in this work by using a specialized function described in section 3.3. The main objective of this work is to reduce the computation complexity as well as the communication complexity in the design of the new key distribution protocol. The computation complexity has been reduced in this work through a ternary tree based key hierarchy which reduces the number of multiplication operations and the optimization in multiplication operations is achieved through the use of the existing Karatsuba fast multiplication technique. Moreover, a specialized function is employed in this work in order to reduce the number of data bits that are transferred between the GC and group members. Several advantages are provided by the proposed algorithm. First, it reduces the computation complexity of the key distribution and updating operations when compared with the existing approaches since it uses a ternary tree based key hierarchy. Secondly, the number of multiplication operations used for key

updating is reduced when compared to other approaches since we use Karatsuba fast multiplication technique. Finally, the proposed algorithm reduces the communication complexity through the use of specialized functions. The remainder of the paper is organized as follows: Section 2 provides a survey of related works. Section 3 discuses the proposed key distribution protocol. Section 4 explains the Ternary tree based key management proposed in this paper. Section 5 performs the comparative performance analysis of the proposed algorithm with the other existing key distribution methods. Section 6 depicts the simulation results. Section 7 gives the concluding remarks and suggests some future directions.

2. Related Work

There are many works on key management and key distribution that was present in the past [1-3] [11] [17-21] [24-26]. In most of the existing key management schemes, different types of group users obtain a new distributed multicast key which is used for encrypting the multimedia data for every session update. Among the various works on key distribution, Maximum Distance Separable (MDS) [5] method focuses on error control coding techniques for distributing re-keying information. The main limitation of this scheme is that it increases both computation and storage complexity. The Data Embedding Scheme proposed in [6] is used to transmit rekeying message by embedding the rekeying information in multimedia data. In that scheme, the computation complexity is O(log n). The storage complexity also increases in the value of O(n) for the server machine and O(log n) for group members. One of the limitations of this scheme is that a new key called embedding key has to be provided to the group members in addition to the original keys, which causes lots of overhead. Level homogeneous key tree [7] based key management scheme was proposed [8] to reduce computation and storage complexity. Key management using key graphs [9] has been proposed by Wong Gouda which consists of creation of secure group and key management graphs scheme using Star based and Tree based method. The limitation of this approach is the high computation cost of group center. A new group keying method that uses one-way functions [10] to compute a tree of keys, called the One-way Function Tree (OFT) algorithm has been proposed by David and Alan. 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, 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 and communication time complexity. ELK [23] uses key derivations based on pseudo-random functions, and the authors also implement recovery mechanisms in their algorithms. It is similar to OFT in the sense that intermediate keys are generated from its children, but Pseudo-Random Functions (PRFs) are used rather than one-way functions.

Wade Trappe and Jie Song proposed a Parametric One Way Function (POWF) [12] based binary tree key Management. The storage complexity is given by $(log_\tau n) + 2$ keys for a group centre. The amount of storage needed by the individual user is given as $\frac{(\tau^{L+1}-1)}{\tau-1}$ Keys where L is the number of levels of the tree. Computation time is represented in terms of amount of multiplication required. The amount of multiplication needed to update the KEKs using bottom up approach is $\tau log_\tau n - 1$. Multiplication needed to update the KEKs using top down approach is $\frac{(\tau-1)log_\tau n(log_\tau n+1)}{2}$. This complexity can be reduced substantially if the numbers of multiplications are reduced. Therefore, in this paper we propose a new Ternary

tree based key Management Scheme that reduces computation time by reducing the number of multiplications required in the existing approaches. We also use a karatsuba fast multiplication algorithm to optimize the multiplication operations used in the key distribution protocol. The proposed method also reduces the amount of information needed to be communicated when there is a change in the group membership. Our proposed algorithm is suitable for single join/leave operation (Single Rekeying operation). When numbers of joining or leaving operations are more, batch joining and leaving operations can be integrated into our proposed key distribution protocol.

3. Proposed Key Distribution Protocol

The proposed framework works in three phases. The first phase is the Group Center initialization, where a multiplicative group is created. In the second phase called Member Initial Join, the members send joining request to the group center and obtain all the necessary keys for participation. The final phase namely the Member Leave phase deals with all the operations to be performed after a member leaves from the group (providing forward secrecy). The proposed framework mainly concentrates on the third phase of Member Leave operation because the computation time is extremely large in providing forward secrecy and is a great challenge in multimedia multicast applications.

3.1 GC Initialization

Initially, the GC selects a large prime number p. This value, p helps in defining a multiplicative group z_p^* and a secure one-way hash function H(.). The defined function, H(.) is a hash function defined from $x \times y \rightarrow z$ where 'x' and 'y' are non-identity elements of z_p^*. Since the function H(.) is a one way hash function, 'x' is computationally difficult to determine from the given function $z = y^x (\mathrm{mod}\ p)$ and 'y'.

3.2 Member Initial Join

Whenever a new user U_i is authorized to join the multicast group for the first time, the user sends a join request to GC. After receiving the joining request, GC verifies it, authenticates the identity of the new user, and randomly generates a new individual secret key k_i and sends it (using a secure unicast) via secure channel which is known only to the user U_i and GC. k_i is a random element in z_p^*. Using this k_i, the Sub Group Keys (SGK) or auxiliary keys and a Group key k_g are given for that user U_i which will be kept in the user U_i database. In order to communicate the SGK and GK to the newly joined user, the group centre encrypts the new SGK and GK using new user's secret key k_i and sends it as a unicast message. Meanwhile, the GC encrypts the new SGK and GK using old SGK and GK and the result is multicast to the already existing users.

3.3 Member Leave

Member Leave operation is completely different from member join operation. In member leave operation, when a member leaves from the group, the GC must avoid the use of old Group key/SGK to encrypt new Group key/SGK. Since old member, knows old GK/SGK, it is necessary to use each user's secret key to perform re-keying operation when a member departs from the services. In the existing key management approaches, this process increases GC's computation time, because the number of multiplication operations to be done in the key updating is more. Therefore, this work aims at reducing the computation time by

decreasing the number multiplication operations to be carried out. Moreover, this work reduces the amount of information to be communicated to the group members to recover the new GK. The following steps describe the key updating process which reduces computation and communication time.
The GC executes the rekeying process in the following steps:

1. GC defines a one way hash function $h(k_i, y)$ where k_i is the i[th] users secret key information, y is the users public key information and computes its value as shown in equation (1).
$$h(k_i, y) = y^{k_i}(mod\ p) \quad (1)$$
2. GC computes the following function $f(k_i, y)$ for the user U_i as shown in equation (2). Next, it computes $f(k_j, y)$ for the user U_j. Similarly, it computes this value for 'n' numbers of users to whom the message has to be sent.
$$f(k_i, y) = \left\lceil \frac{h(k_i, y)}{k_i} \right\rceil \quad (2)$$
3. It defines a new function $g(k_i, y)$ which is obtained by adding $f(k_i, y)$ with the value of 'p' as shown in equation (3).
$$g(k_i, y) = p + f\ (k_i, y) \quad (3)$$
The purpose of adding the value of 'p' with $f(k_i, y)$ is to allow each user to recover the original keying information. This function has been introduced newly in this proposed key distribution algorithm to reduce the computation and communication time.
4. GC computes (encrypts) the new keying information $\gamma_g(t)$ for the new group key, $K_g \in z_p^*$ which is generated at time 't' and that has to be sent to the group members as shown below:
$$\gamma_g(t) = K_g\ (t) + \prod_{i=1}^{n-1}\left(g(k_i, y)\right) \quad (4)$$
5. GC sends the newly encrypted key $\gamma_g(t)$ value to the existing group members.

Upon receiving the encrypted information from the GC, an authorized user U_i of the current group executes the following steps to obtain the new group key.

a. Calculate the value $h(k_i, y) = y^{k_i}(mod\ p)$ where k_i is user's secret key and 'y' is the old group keying information which is known to all the existing users. If 'y' is a random value selected from the group, it is sent as a broadcast message to the existing group members.
b. Compute $f(k_i, y) = \left\lceil \frac{y^{k_i}(mod\ p)}{k_i} \right\rceil$
c. Add 'p' value with the function $f(k_i, y)$.
d. A legitimate user U_i can decode the rekeying information to get the new group key $K_g(t)$ by calculating the following value:
$$\gamma_g(t) mod\ \left(g(k_i, y)\right) \quad (5)$$

Since (i) $K_g(t) \in z_p^*$ and $K_g(t) < p < p + f\ (k_i, y)$,
(ii) $\prod_{i=1}^{n-1}\left(g(k_i, y)\right)$ is divisible by $g(k_i, y)$

$$\gamma_g(t) mod\ \left(g(k_i, y)\right) = K_g\ (t)$$

The proposed method adds the p value with $f(k_i, y)$ to reduce the computation and communication time. The '+' sign in equation (3) is the addition operation which plays a vital role in reducing the computation time complexity. In the existing approaches [1][12] concatenation operator was used to concatenate 1 or some other value in front of the function $f(k_i, y)$. This may increase the number of digits produced by the function $g(k_i, y)$.

Therefore, the computation time increases when performing the multiplication operation as the number of digits produced by the function $g(k_i, y)$ increases. If the output produced by the function is an 'n' digit number, in general, to multiply two 'n' digit numbers requires n^2 multiplications. Hence, the computation time can be reduced if the output produced by the function $g(k_i, y)$ is reduced compared with the existing approaches. In order to optimize the multiplication operation used in rekeying message as shown in equation (4), Karatsuba Divide and conquer multiplication method is used as an optimization method in our proposed key distribution protocol. The working of the optimization method is explained in the following section.

3.4 Optimization Method

Assumption:

Let $g(k_i, y)$ be a multiplication function, where k_i=secret key of a user, y= old group key. Now, 'σ_i' is the size of the output digit produced from the function $g(k_i, y)$ for U_i number of users where i=1, 2, 3,..., n (n= size of the group).

Example:

Consider a scenario, where σ_1=2 and σ_2=2. A set of 'σ_i' are multiplied to form the rekeying message as shown in equation (4) using traditional multiplication operation in most of the existing key distribution approaches [4][12]. In order to multiply σ_1 and σ_2 using traditional multiplication operation, the algorithm takes 4 numbers of multiplications. In general, when two 'σ' digit numbers are to be multiplied, it takes (σ^2) multiplication operations in order to obtain the solution.

For optimizing the number of multiplication operations, Karatsuba fast multiplication divide and conquer approach [13][14][15][16] is used in our proposed key distribution algorithm. The number of multiplication operations to be performed in total to obtain the solution for the 'σ' digit number will be ($\sigma^{1.585}$). If σ is a long digit number, it divides the number in to tow halves and those products can be multiplied by recursive calls of the Karatsuba divide and conquer algorithm. The recursion can be applied until the numbers are so small that they can (or must) be computed directly. We can obtain the whole product used in the function $g(k_i, y)$ by using "divide and conquer" method recursively. Hence, the multiplication time complexity is given by $O(\sigma^{\log_2 3})$. Therefore, it is faster than the classical algorithm, which requires σ^2 single-digit products and the complexity is $O(\sigma^{\log_2 4})$. Karatsuba fast multiplication approach works well when the value of σ >16 digits. However, if the number of digits of σ <16, this algorithm shall not show a significant difference. In order to make the optimized use of karatsuba fast multiplication approach, the group size in our proposed key distribution algorithm can have a 16-ditigt, 32-digit, 64-digit and 128-digit, etc. In the proposed algorithm given in section 3.3, we have analyzed and tested the algorithm for a multiplicative group size p as 16-digit, 32-digit and 64-digit prime numbers. The key values used in our algorithm are of 16 and 32 digit numbers.

Theorem 1: The number of multiplication in group key computation is in the order of $O(\sigma^{1.585})$ when Karatsuba divide and conquer multiplication [2] is employed for key computation where the key size is a σ digit number.

Proof: Initially, divide the input number σ in to three $\frac{\sigma}{2}$ digit numbers each can take three multiplications of $\frac{\sigma}{2}$ digits, which is represented as $\frac{\sigma}{2} \times 3$. Break each of the resulting

numbers further into $\frac{\sigma}{4}$ digit parts and perform the multiplications with these parts. Repeat the same steps until single digits are obtained for the input numbers in the level $\log_2\sigma$. Simplifying the above statements, the following formulations are obtained.

$$= \sigma + \frac{\sigma}{2} \times 3 + \frac{\sigma}{4} \times 3^2 + \frac{\sigma}{8} \times 3^3 + \cdots + \frac{\sigma}{2^{\log_2\sigma}} \times 3^{\log_2\sigma}$$

$$= \sigma + \frac{3}{2} \times \sigma + \frac{3^2}{2^2} \times \sigma + \frac{3^3}{2^3} \times \sigma + \cdots + \frac{3^{\log_2\sigma}}{2^{\log_2\sigma}} \times \sigma$$

$$= \sigma + \frac{3}{2} \times \sigma + \left(\frac{3}{2}\right)^2 \times \sigma + \left(\frac{3}{2}\right)^3 \times \sigma + \cdots + \left(\frac{3}{2}\right)^{\log_2\sigma} \times \sigma$$

$$= \sigma\left(1 + \frac{3}{2} + \left(\frac{3}{2}\right)^2 + \left(\frac{3}{2}\right)^3 + \cdots + \left(\frac{3}{2}\right)^{\log_2\sigma}\right)$$

This series is in the form of finite geometric series $1 + \omega + \omega^2 + \cdots + \omega^r$ and this can be written as,

$$1 + \omega + \omega^2 + \cdots + \omega^r = \frac{\omega^{r+1}-1}{\omega-1}$$

Substitute $\omega = \frac{3}{2}$ to simplify this equation.

$$= \sigma\left(\frac{\left(\frac{3}{2}^{\log_2\sigma+1}-1\right)}{\left(\frac{3}{2}-1\right)}\right) = \sigma\left(\frac{\left(\frac{3}{2}^{\log_2\sigma+1}-1\right)}{\left(\frac{1}{2}\right)}\right) = 2\sigma\left(\frac{3}{2}^{\log_2\sigma+1} - 1\right)$$

$$= 2\sigma\left(\left(\frac{3}{2}\right)^{\log_2\sigma}\left(\frac{3}{2}\right)^1 - 1\right) = 2\sigma\left(\frac{3}{2}\right)^{\log_2\sigma}\left(\frac{3}{2}\right)^1 - 2\sigma$$

$$= 3\sigma\left(\frac{3}{2}\right)^{\log_2\sigma} - 2\sigma \quad = 3\sigma\left(\frac{3^{\log_2\sigma}}{2^{\log_2\sigma}}\right) - 2\sigma = 3\sigma\left(\frac{3^{\log_2\sigma}}{\sigma}\right) - 2\sigma$$

$$= 3\sigma^{\log_2 3} - 2\sigma \quad = O(\sigma^{1.585})$$

Table 1. Computation Time measured for various key sizes

Prime number size (in digits)	Private key/GK size (in digits)	Number of Multiplications	Existing Binary tree approach Key Computation Time (ns)	Optimized KKD Key Computation Time(ns) (Proposed)
16	8	2	1787059	1585798
16	8	3	2897065	2695804
16	16	2	1926072	1748828
16	16	3	3567098	3389854
32	16	2	2543346	2367102
32	16	2	4059594	3892350
32	32	2	4203738	4062044
32	32	3	8200456	8058762

Table 1 shows the computation time measured from the optimized key distribution protocol that employs Karatsuba divide and conquer multiplication algorithm. From the table, it is evident that the optimized key distribution algorithm takes less computation time for the key distribution in comparison with the other algorithms given in the literature. The sizes of the key values are taken as16 and 32 digits for the experimental analysis. However, from the table it is also evident that when the key value is an 8-digit number the computation time is close to the values of the traditional multiplication used in other existing algorithms. In conclusion, the proposed optimized key distribution algorithm performs best when the keys are of larger size (>16 digits).

4. Ternary Tree based Approach

Scalability can be provided by employing the proposed approach in a key tree based key management scheme to update the GK and SGK. **Fig. 1** shows a key tree in which, the root is the group key, leaf nodes are individual keys, and the other nodes are auxiliary keys (SGK). A key tree is a hierarchical arrangement of a set of keys where each node of a key tree contains a key called as k-nodes. Auxiliary keys are assigned to internal k-nodes, and users secret keys are placed in the leaf level k-nodes. Each leaf level k-node is associated with a unique user node u-node, which represents a particular member in the group. An individual member's secret key is only shared by the server and the corresponding member, and an auxiliary key is shared by members that belong to the subtree rooted at the k-node of the auxiliary key. An individual user's secret key is established at the time a member joins the group, and it remains valid until the member leaves. On the other hand, auxiliary keys are frequently updated and sent to members to keep multicast communication secrecy. For Example, if a member M_9 from **Fig. 1** leaves from the group, the keys on the path of his/her leaf node to the tree's root should be changed. Hence, only the keys $k_{7,9}$ and $k_{1,9}$ will become invalid.

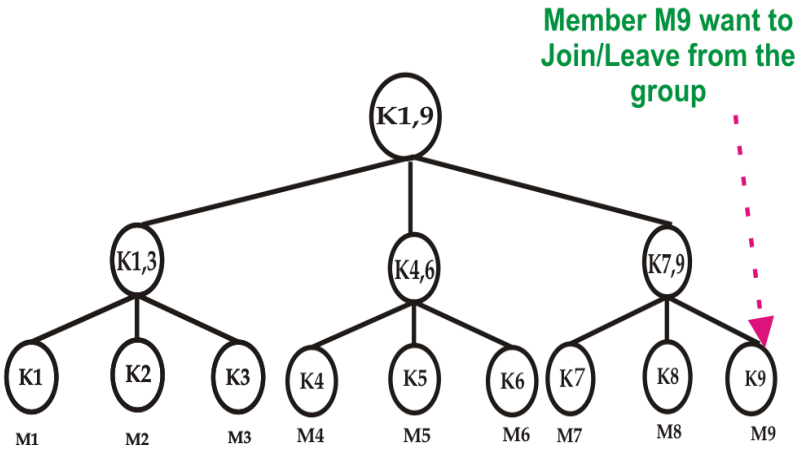

Fig. 1. Key Tree based Key management Scheme

Therefore, these keys must be updated. In order to update the keys, two approaches namely top-down and bottom-up are used in the members departure (Leave) operation. In the top-down approach, keys are updated from root node to leaf node. On the contrary, in the bottom-up approach, the keys are updated from leaf node to root node. When member M_9 leaves from the group, GC will start to update the keys $k_{7,9}$ and $k_{1,9}$ using bottom-up approach. In top-down approach, the keys are updated in the order $k_{1,9}$ and $k_{7,9}$. The number of multiplications required to perform the rekeying operation is high in top down approach than bottom up approach. This is evident from the complexities shown in equation (6) and (7). In the ternary tree based method, the number of multiplications needed to compute the re-keying information using the bottom up approach is given by the relation,

$$B = \tau + 2(log_{\tau}n - 2) + log_{\tau}n \quad (6)$$

where,

B= The number of multiplications needed in bottom up approach, τ = The number of children (degree) for each node of the tree, $log_{\tau}n$ = L=Level of the tree (0 …v), and n = number of registered users. Similarly, the number of multiplications needed for top down approach in the ternary based method to compute the re-keying information can be given by

the relation,

$$T = log_{\tau}n(1 + log_{\tau}n) \quad (7)$$

Table 2 summarizes the overall computation complexity of our proposed Karatsuba multiplication based Key Distribution protocol named as KKD with most of the existing well known key distribution protocols SKDC [9], OFT [10], ELK [23] and LKH [24]. The notations used for comparisons are defined as: n is the number of members, k is size of the key used in various algorithms, τ is the maximum number of children of each node of the tree, p is the size of the prime number used to define the multiplicative group. The length of a path of the key tree of LKH, OFT, ELK and SKDC is h, in particular, h_2 for binary key trees, h_{τ} for τ -ary key trees, and h_{τ}=$[log_{\tau}n]$ for balanced key trees.

E, D, ED, F, R, M, Ap, A denote the computation costs of encryption, decryption, erasure decoding, key derivation, random key generation, modulo division, appending and addition respectively. Erasure decoding operations for an MDS code needs $O(n^2)$ arithmetic operations if standard erasure decoding algorithms are used. Even if the Fast decoding algorithms [22] are used they need $O(nlogn)$ operations in the receiver side. Similarly, the time taken to perform a decryption, random key generation operation used in the receiver side in SKDC, OFT, ELK and LKH is high compared with performing simple addition and modulo division used in our KKD approach. Considering the computation costs of LKH, OFT, ELK, MDS, Binary and SKDC our proposed approach KKD uses only a simple addition and modulo division operations in the user side during the key updating process. Therefore, the proposed KKD multicast key distribution takes less computation power in the user side compared with some of the existing key distribution protocols. Moreover, for optimizing the computation power of GC we have incorporated the existing karatsuba multiplication when the key updating operation is used in the GC. This improves the group center computation cost slightly compared with existing approaches.

4.1 Security Analysis

This work analyzes the proposed protocol for forward secrecy and backward secrecy. The assumption of the implemented protocol is that an adversary might be a group member sometime. Backward secrecy is the technique of preventing a new member from accessing the communication sent before joining the group. In order to access the previous communication, an adversary needs to make the previous group key. Moreover, if the adversary becomes a group member, it may try to derive the previous group key which is not permitted. In the proposed protocol, the newly updated group key is encrypted with an old group key when it is communicated to old group members, an adversary needs to find the old group key. For performing the encryption operation explained earlier, the traditional DES encryption algorithm is used in the join procedure of the protocol. This property makes the situation impossible for the adversary to compute the previous group key. Consequently, the adversary cannot access the communication sent before join, which means the proposed approach supports the initial security requirement.

Forward secrecy is the technique of preventing an old member from accessing current communication after leave operation. When a member leaves the group, he/she may try to derive the group or subgroup key. In the proposed protocol, it is impossible for an adversary to compute the current group key after the leave for the same reasons that was explained for the backward secrecy technique. Because when a member leaves from the group, GC generates a polynomial using all the known keys of the existing users and adds the new group key to this polynomial. The user can then divide this polynomial using his/her personal keying information. Therefore, a user who had already left from the service cannot

find the new group key since his/her personal keying information is not included in the polynomial. However, it is to be noticed that even after a member is left from the service, the departing member can try to find any one of the existing member's secret in order to obtain the newly communicated group key by guessing out all the keys of the multiplicative group using brute force attack. In this case, if the size of k_i is w bits, then the attacker has to use the total number of trial of 2^{w}. The time taken to derive k_i can be increased by choosing a large k_i for each user's secret key. In this work, the size of k_i must be 64bits and prior experiments were conducted with 32 bits. If the time taken to perform a single attempt using brute force attack is $1\ \mu s$, then the total time required will be $2^{63}\mu s$ = 292471 years. Consequently, an adversary cannot find the group key in order to access the current communication, which means the second security requirement is supported in this protocol.

Table 2. Computation, Storage and Communication Complexities of Various Key Distribution Schemes

	MDS	Binary	OFT	ELK	LKH	SKDC	KKD (Proposed)
Computation Cost (GC)	$l_r + (n-1)m$	$\tau \times h_2 - 1$	$h_2(E + 2F)$	$h_2(2E + 7F)$	$h_\tau(\tau E + R)$	$(n-1)(E+R)$	$\tau + 2(h_\tau - 2) + h_\tau$
Computation Cost(User)	$(h_\tau - 1)(H + ED)$	$(h_2 - 1)(M + Ap)$	$h_2(D + F)$	$h_2(D + 4F)$	$h_\tau(D)$	D	$(h_\tau - 1)(M + A)$
Storage Complexity (user)	$h_\tau + t$	$h_2 + 2$	(h_2)	(h_2)	(h_τ)	2	$h_\tau + 1$
Storage Complexity (GC)	$\frac{(\tau n - 1)}{(\tau - 1)}$	$\frac{(\tau^{h_2+1} - 1)}{\tau - 1}$	$2n - 1$	$2n - 1$	$\frac{(\tau n - 1)}{(\tau - 1)}$	$n + 1$	$\frac{(\tau n - 1)}{(\tau - 1)}$
Communication Complexity	$\tau \times h_\tau$	$\frac{\tau \times h_2}{(\tau \times K) + (\tau \times h_2)}$	$h_2 \times K$	$(2h_2 \times K)$	$(\tau h_\tau \times K)$	$(n-1) \times K$	$(h_\tau - 1)[(3 \times \mathrm{p}) - 1] + 2\mathrm{p}$

5. Performance Evaluation

In this section, we study the performance of our proposed algorithms and compare them with the Euler's totient based Algorithm, MDS algorithm, secure group communication, one way function tree, Binary tree based, Efficient large group key distribution, and Logical key hierarchy Algorithm described in [1][5][9][10][12][23][24] which are labeled as ETF, MDS, SKDC, OFT, Binary, ELK and LKH respectively. We consider two performance metrics such as Computation Time and Communication Time. Computation Time is defined as the time taken to compute the group key at the Group Center area and the Group Members' area. Communication Time is defined as the time taken to receive the keying material from the Group Centers area. For comparing the computation time of the proposed algorithm with seven different algorithms present in the literature, the key size is taken to be 8-digit, 16-digit and 32-digit values. **Table 3** and **Table 4** shows the measured computation time in

nanoseconds for such comparisons. It is evident from the values that the computation time of our proposed algorithm is found to be better both in the server area and the client area than the other algorithms. The proposed method adds p value with $f(k_i, y)$ instead of appending a number in front of it to reduce the number of key information bits communicated from GC to group members. The amount of information bits needs to be communicated while updating the keys for our proposed approach and existing approach are calculated and summarized in **Table 5**. Since this key management scheme uses a ternary tree based key tree to store all the keys, the maximum number of multiplication operations performed to update a single key is two at each level except the last but one level. Hence, if a tree has maximum of N children under each node then N-1 multiplications are performed to update a particular key of a key tree when a user join/leave from the multicast service. As the amount of multiplications increase, the number of bits that are to be communicated in our proposed approach is reduced gradually and hence it reduces the overall communication delay.

Table 3. Server Group key Computation Time for various Key Distribution Approaches

No. of Multiplications	Public key/Old GK	Private Keys	Size of the Prime Number (in digits)	New GK Size (in digits)	Binary (ns)	ETF(ns)	MDS (ns)	OFT (ns)	ELK (ns)	LHK (ns)	SKDC (ns)	KKD (ns) (Proposed)
2	65236	32067, 92038	8	5	753408	211948432	599231	609381	624495	613495	627495	552147
2	3405833	634090, 5609432	8	6	846185	74401457	760156	702347	717272	706272	720272	644924
2	99436	76420, 13042	16	5	876905	108734791	790874	745116	757992	746992	760992	685644
2	5325021	630229, 5329022	16	6	976493	87838705053	878956	843730	857580	846580	860580	785232
3	52354	34602, 65708, 34503	8	5	894126	66234234	832145	872947	886348	875348	889348	814000
3	7239530	2135082, 5592011, 9338043	8	6	1027245	66173260	965623	1006572	1019467	1008467	1022467	947119
3	21321	67702, 57524, 13456	16	5	1079355	134731825	945647	940672	950442	939442	953442	878094
3	4213531	4562404, 7429042, 3465632	16	6	1316729	1317281552	1162346	1178946	1187816	1176816	1190816	1115468

From the experiments carried out in this work, it has been observed that when the group consists of 243 users and if one user wants to leave from a ternary tree, then the number of digits to be communicated for updating a single key from any level other than $l-1$ th level using proposed approach is 47 digits, if the prime number p value is set as a 16 digit number which is equal to 64bits. Similarly, for the same p value the number of digits to be communicated becomes 49 digits in the existing approach and hence it increases 8-bits for updating a single key. If the ternary tree has 5 levels (total number of users=243) then the total number digits need to be communicated to update the group key using our proposed approach is represented as 47+47+47+47+32=220digits. However, in the existing schemes [1][12] the communication cost is computed as 49+49+49+49+33=229digits for a ternary tree where l =5. This gives us the overall reduction of 9 digits=36bits in our proposed

approach. In general, if the degree of the multiplicative group z_p^* used in our approach is p, then the amount of keying information to be communicated for updating each intermediate key (auxiliary key) is denoted as [(3×p)-1] except for the level $l-1$ where leaving/joining operation takes place. So the communication cost takes to update all the keys from the leaf to the root is given by,

$$CC_{exist} = (log_\tau n - 1)[(3 \times p) + 1] + (2p + 1) \quad (8)$$

Where,

CC_{exist} = Communication Cost of existing key management approach,
p = degree of the multiplicative group,
τ = degree of the tree and n = Total number of users.

Table 4. Client Group Key Computation time for Various Key Distribution Approaches

No. of Multiplications	Public key/Old GK	Private Keys	Size of the Prime Number (in digits)	New GK Size (in digits)	Binary (ns)	ETF(ns)	MDS (ns)	OFT (ns)	ELK (ns)	LHK (ns)	SKDC (ns)	KKD (ns) (Proposed)
2	65236	32067, 92038	8	5	520372	87477763	494678	335617	391617	348617	319111	314605
2	3405833	6340902, 5609432	8	6	588792	62202985	637209	404037	460037	417037	387531	383025
2	99436	76420, 13042	16	5	640457	87105870	569879	465702	521702	478702	449196	444690
2	5325021	6302293,5329022	16	6	433325	86853329641	397543	258570	314570	271570	242064	237558
3	52354	34602,65708, 34503	8	5	526888	58699556	578213	463268	519268	476268	446762	442256
3	7239530	2135082, 5592011, 9338043	8	6	450018	61553685	431432	386398	442398	399398	369892	365386
3	21321	67702, 57524, 13456	16	5	897951	154361288	744304	713196	769196	726196	696690	692184
3	4213531	4562404, 7429042, 3465632	16	6	987455	1244607297	913456	802700	858700	815700	786194	781688

$$CC_{prop} = (log_\tau n - 1)[(3 \times p) - 1] + 2p \quad (9)$$

Where,

CC_{prop} = Communication Cost of proposed key management approach,
p = degree of the multiplicative group,
n = Total number of users,
τ = degree of the tree.

Table. 5 Length of the Rekeying Messages for various Key Distribution Approaches

No. of Multiplications	Public key/Old GK	Private Keys	Length of Prime Number(in digits)	Group Key Size(in digits)	Binary Tree (in digits)	ETF (in digits)	MDS (in digits)	KKD (in digits) (Proposed)
2	65236	32067,92038	8	5	17	17	16	16
2	3405833	6340902,5609432	8	6	17	17	16	16
2	99436	76420,13042	16	5	33	31	32	32
2	5325021	6302293,5329022	16	6	33	31	32	32
3	52354	34602,65708,34503	8	5	25	25	24	24
3	7239530	2135082,5592011,9338043	8	6	25	25	24	24
3	21321	67702,57524,13456	16	5	49	48	48	47
3	4213531	4562404,7429042,3465632	16	6	49	49	48	47

Table 6. Communication Delay for various Key Distribution approaches

No. of Multiplications	Public key/old GK	Private Keys	Binary Tree (ns)	ETF(ns)	MDS (ns)	ELK (ns)	LHK (ns)	SKDC (ns)	OFT (ns)	KKD (ns) (Proposed)
2	65236	32067,92038	1098458	868635	974632	671940	771940	1171940	506873	506773
2	3405833	6340902,5609432	498030	375151	512546	437103	537103	937103	272036	172036
2	99436	76420,13042	640457	298495	640023	507243	607243	1007243	342176	343176
2	5325021	6302293,5329022	433325	149545	423040	450312	550312	950312	285245	284245
3	52354	34602,65708,34503	419834	296490	417653	349670	449670	849670	184603	183603
3	7239530	2135082,5592011,9338043	350018	343275	326578	407737	507737	907737	242670	232670
3	21321	67702,57524,13456	532463	510320	504567	608166	708166	1108166	443099	433099
3	4213531	4562404,7429042,3465632	476145	496499	464343	406979	506979	906979	241912	141912

Similarly, the analysis for the Communication Time is also done and the results are tabulated in **Table 6**. From the result analysis, it is clear that the communication time taken by our proposed algorithm is found better than the other algorithms.

6. Simulation Results

The proposed method has been simulated in Java (in a P4 machine with 2GB RAM) for more than 6000 users and we have analyzed the computation time with existing approaches to perform the rekeying operation.

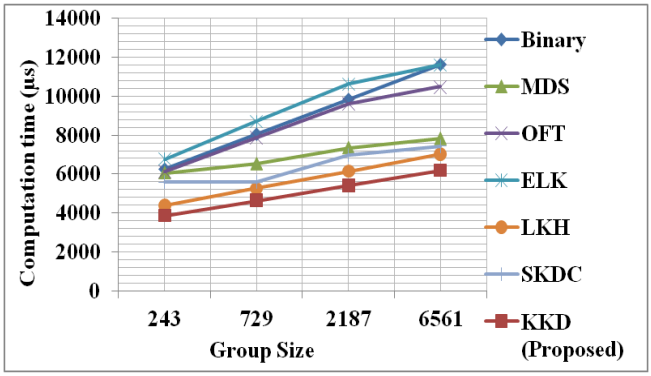

Fig. 2. GC Key Computation Time of various Key Distribution schemes

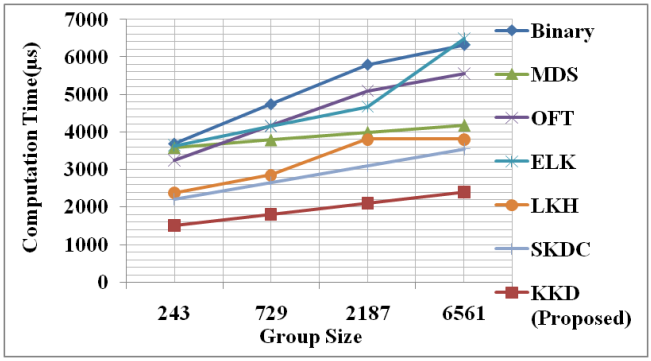

Fig. 3. User Key Computation for getting the new key value

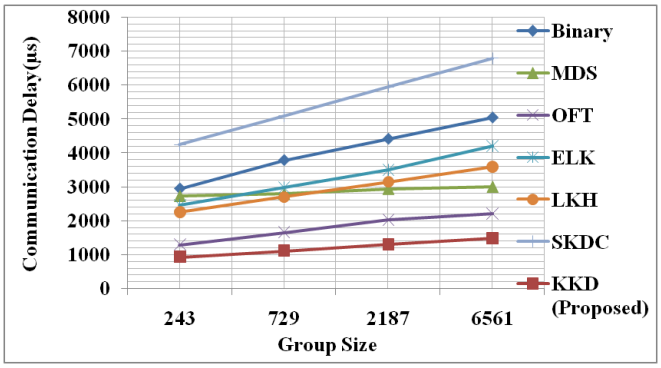

Fig. 4. Communication Delay of various Key Distribution schemes

The graphical results shown in **Fig. 2** are used to compare the GC key computation time of the proposed method with the existing methods. It compares the results obtained from our proposed KKD with MDS, SKDC, OFT, Binary, ELK and LKH. From **Fig. 2,** it is observed that when the group size is 6561, the key computation time is found to be 6181.152µs in our proposed approach for updating all the keys from the leaf node to the root node where the key size=16 digits, which is better in comparison with the other existing schemes. Moreover, if the number of members who are joining and leaving increases the computation time proportionately increases. However, it is less in comparison with the existing approaches. The results shown in **Fig. 3** are used to compare the user's key computation time of our proposed method with the existing methods.

It compares the results obtained from the KKD scheme with existing approaches and it is observed that when the group size is 6561, the key recovery time of a user found to be 2405.432µs in our proposed approach, which is better in comparison with the other existing schemes. The results illustrated in **Fig. 4** shows the communication time taken to receive all the rekeying information from leaf nodes to the root node in order to update the group key. Our proposed approach (KKD) shows a reduced communication delay when compared to other algorithms, which is approximately 1476.824µs.

7. Conclusion

In this paper, new algorithms have been proposed to tackle challenges exist in secure multimedia multicast. Therefore, a new ternary tree-based key distribution protocol (key value = 'n' bit numbers) has been proposed to provide secure multicast communications. The proposed algorithm has two dimensional focuses namely Minimal computation complexity and minimal communication overhead. When the key size is small (key size=8 digits) the computation time decreases by 0.2 milliseconds and when the key size increases (16 or 32 digits), the computation time decreases by 0.15-0.18 milliseconds for updating a single key from any level of the ternary tree. With respect to the communication time, the communication delay is reduced in this work by 0.15-0.23 milliseconds when the key size ranges from 8-digits to 16-digits. Further extensions to this work are to devise techniques for reducing the storage complexity which is the amount of storage required to store the key related information, both in GC and group members area.

References

[1] Vijayakumar,P., Bose,S., Kannan,A., and Siva Subramanian,S., "A secure key distribution protocol for multicast communication," Balasubramaniam P. (eds.) Gandhigram-India, CCIS.,Vol.140, Springer, Heidelberg, 2011, pp. 249-257. Article (CrossRef Link)

[2] Vijayakumar,P., Bose,S., Kannan,A., "Centralized Key Distribution Protocol using the Greatest Common Divisor Method," *Computers & Mathematics with Applications, Elsevier*, Article Online. Article (CrossRef Link)

[3] Mingyan Li, Poovendran,R., David McGrew,A., "Minimizing center key storage in hybrid one-way function based group key management with communication constraints," Elsevier, *Information Processing Letters*, Vol.93, No.4, pp. 191-198, 2004. Article (CrossRef Link)

[4] Patrick P.C Lee, John C. S Lui, David K.Y Yau, "Distributed collaborative key agreement protocols for dynamic peer groups," in *Proc. of the IEEE International Conference on Network Protocols*, pp. 322, 2002. Article (CrossRef Link)

[5] Lihao Xu, Cheng Huang, "Computation-efficient multicast key distribution," *IEEE Transactions on Parallel and Distributed Systems*, Vol.19,No.5, pp. 1-10,2008. Article (CrossRef Link)

[6] Wade Trappe, Jie Song, Radha Poovendran, K. J Ray Liu, "Key distribution for secure multimedia multicasts via data embedding," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 1449-1452, 2001. Article (CrossRef Link)

[7] Lee,J.S., Son,J.H., Park, Y.H.,Seo,S.W., "Optimal level-homogeneous tree structure for logical key hierarchy," in *Proc. of IEEE Conference on Communication System Software and Middleware Workshop (COMSWARE)*, 2008. Article (CrossRef Link)

[8] Dong-Hyun Je, Jun-Sik Lee, Yongsuk Park, Seung-Woo Seo, "Computation-and-storage-efficient key tree management protocol for secure multicast communications," *Elsevier, Computer Communications, Vol.33*, No.6, pp. 136-148, 2010. Article (CrossRef Link)

[9] Wong, C., Gouda,M., and Lam,S., "Secure group communications using key graphs, *IEEE/ACM Transactions on Networking*," Vol. 8, No.1, pp.16-30, 2000. Article (CrossRef Link)

[10] David,A., McGrew and Alan T. Sherman, "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions on software engineering*, Vol.29,No.5, pp. 444-458, 2003. Article (CrossRef Link)

[11] Poovendran,R., Baras,J.S., "An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes," *IEEE Transactions on Information Theory*, Vol. 47, No.7, pp. 2824–2834, 2001. Article (CrossRef Link)

[12] Wade Trappe, Jie Song, Radha Poovendran, Ray Liu K J, "Key management and distribution for secure multimedia multicast," *IEEE Transactions on Multimedia*,Vol.5,No.4, pp. 544-557, 2003. Article (CrossRef Link)

[13] Xianjin Fang, Longshu Li, "On karatsuba multiplication algorithm," in *Proc. of 7th Int. Data, Privacy, and E-Commerce Symp., Washington, DC, USA*, pp. 274-276, 2007. Article (CrossRef Link)

[14] Tom St Denis, "Bignum math implementing cryptographic multiple precision arithmetic," *SYNGRESS Publishing*, 2003. Article (CrossRef Link)

[15] Meftah,S., Razali,R., Samsudin,A., Budiator,R., "Enhancing public-key cryptosystem using parallel karatsuba algorithm with socket programming," in *Proc. of 9th Asia-Pacific Conference on Coomunication, Penang, Malaysia, September 21-24*, pp. 706-710, 2003. Article (CrossRef Link)

[16] Jebelean,T., "Using the parallel karatsuba algorithm for long integer multiplication and division," in *Proc. of European Conference on Parallel Processing, in Lecture Notes in Computer Science*, pp. 1169-1172, 1997. Article (CrossRef Link)

[17] Vijayakumar,P., Bose,S., Kannan,A., and Siva Subramanian,S., "An effective key distribution protocol for secure multicast communication," in *Proc. of IEEE International Conference on Advanced Computing, Chennai*, December 14-16, pp. 102-107, 2010. Article (CrossRef Link)

[18] Sandeep S. Kulkarni, Bezawada Bruhadeshwar, "Key-update distribution in secure group communication," Elsevier, *Computer Communications*,Vol. 33,No.6, pp.689-705, 2010. Article (CrossRef Link)

[19] Bezawada Bruhadeshwar, Kishore Kothapalli, "A family of collusion resistant symmetric key protocols for authentication," *ICDCN*, pp. 387-392, 2008. Article (CrossRef Link)

[20] Bezawada Bruhadeshwar, Kishore Kothapalli, Maddi Sree Deepya, "Reducing the cost of session key establishment," *ARES*, pp. 369-373, 2009. Article (CrossRef Link)

[21] Bezawada Bruhadeshwar, Kishore Kothapalli, Poornima,M., Divya,M., "Routing protocol security using symmetric key based techniques," *ARES*, pp. 193-200, 2009. Article (CrossRef Link)

[22] MacWilliams,F.J., and Sloane,N.J.A., "The theory of error correcting codes," North-Holland Math. Library,1977. Article (CrossRef Link)

[23] Adrian Perrig, Dawn Song, Tygar,J.D., "ELK: a new protocol for efficient large-group key distribution," in *Proc. of IEEE Symposium on Security and Privacy Symposium*, pp. 247–262, 2001. Article (CrossRef Link)

[24] Harney,H., Harder,E., "Logical key hierarchy protocol," Internat Draft, IETF, Expired in August 1999. Article (CrossRef Link)

[25] Xixiang Lv and Hui Li, "Group Key Agreement From Signcryption," *KSII transactions on internet and information systems*, Vol. 6, No. 12, pp. 3338-3351, Dec 2012. Article (CrossRef Link)

[26] Guiyi Wei, Xianbo Yang and Jun Shao, "Efficient Certificateless Authenticated Asymmetric Group Key Agreement Protocol," *KSII transactions on internet and information systems*, Vol. 6, No.12, pp.3352-3365, Dec 2012. Article (CrossRef Link)

P.VijayaKumar completed his Ph.D in Computer Science and Engineering in Anna University Chennai in the year 2013. He Completed Master of Engineering in the field of Computer Science and Engineering in Karunya Institute of Technology affiliated under Anna University Chennai in the year 2005. He completed his Bachelor of Engineering under Madurai Kamarajar University, Madurai in the year 2002. He was working as a Senior lecturer in All Nations University Ghana, West Africa from 2007 to 2008. He is presently working as an Assistant Professor at Anna University Chennai (University College of Engineering, Tindivanam), Chennai, India. His main thrust research areas are Key management in Network Security and Multicasting in Computer Networks.

S.Bose has completed his Master of Engineering in the field of Computer Science and Engineering in the year 1996, affiliated under Madurai Kamarajar University, Madurai. After completion, he was working as a lecturer in Anna University Chennai, India. He received the doctoral program (Ph.D.) degree in Computer Science and Engineering from Anna University Chennai, India in the year 2007. Henceforth, he is working as an Assistant Professor in the Department of Computer Science and Engineering at Anna University Chennai. He has done various projects in the area of Computer and Network Security and he has published large numbers of paper in the field of Intrusion Detection Systems.

A. Kannan completed his Master of Engineering in the field of Computer Science and Engineering in Anna University Chennai, India in the year 1991. After completion, he was working as an Assistant Professor in Anna University Chennai, India. He received his Ph.D. degree in Computer Science and engineering from Anna University Chennai, India in the year 2000. He is presently working as a Professor in the Department of Computer Science and Engineering, Anna University Chennai, Chennai. He is also working as a Deputy Director for distance education programs. He has successfully produced sixteen PhD candidates. He has published more than hundred papers in various fields of several reputed journals like Elsevier, Springer, IET, etc. His main thrust areas of interests are Artificial Intelligence and Data Base Management Systems.

L.Jegatha Deboarh completed her Ph.D in Computer Science and Engineering in Anna University Chennai in the year 2013 and completed her Master of Engineering in the field of Computer Science and Engineering in Karunya Institute of Technology affiliated under Anna University Chennai in the year 2005. She completed her Bachelor of Engineering under Madurai Kamarajar University, Madurai in the year 2002. She was working as a Senior lecturer in All Nations University Ghana, West Africa from 2007 to 2008. She is presently working as an Assistant Professor at Anna University Chennai (University College of Engineering, Tindivanam), Chennai, India.