www.ietdl.org

Published in IET Information Security
Received on 15th November 2012
Revised on 13th March 2013
Accepted on 3rd April 2013
doi: 10.1049/iet-ifs.2012.0352

ISSN 1751-8709

Chinese remainder Theorem based centralised group key management for secure multicast communication

Pandi Vijayakumar[1], *Sudan Bose*[2], *Arputharaj Kannan*[3]

[1]*Department of Computer Science and Engineering, University College of Engineering Tindivanam, Melpakkam, Tindivanam, Tamil Nadu-604 001, India*
[2]*Department of Computer Science and Engineering, Anna University Chennai, Chennai, Tamil Nadu-600 025, India*
[3]*Department of Information Science and Technology, Anna University Chennai, Chennai, Tamil Nadu-600 025, India*
E-mail: vijibond2000@gmail.com

Abstract: Designing a centralised group key management with minimal computation complexity to support dynamic secure multicast communication is a challenging issue in secure multimedia multicast. In this study, the authors propose a Chinese remainder theorem-based group key management scheme that drastically reduces computation complexity of the key server. The computation complexity of key server is reduced to $O(1)$ in this proposed algorithm. Moreover, the computation complexity of group member is also minimised by performing one modulo division operation when a user join or leave operation is performed in a multicast group. The proposed algorithm has been implemented and tested using a key-star-based key management scheme and has been observed that this proposed algorithm reduces the computation complexity significantly.

1 Introduction

In multicast communication, messages are sent from one sender to a group of members. Multicast group formation and group communication are very common in the internet scenario. This is helpful for sending and exchanging private messages among group members. Moreover, multimedia services such as pay-per-view, video conferences, sporting events, audio and video broadcasting are based on multicast communication where multimedia data are sent to a group of members. In order to form groups and to control the activities of a group, a leader in the group is necessary. In most of the multimedia group communication, a group centre (GC) or key server (KS) is responsible for interacting with the group members and also to control them. In such a scenario, the groups participating in the communication can be either opened or closed with regards to senders. In a closed group, only registered members can send messages to their group. In contrast, data from any sender are forwarded to the group members in open groups.

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. Therefore, the static approach distributes an unchanged group key (GK) to the members of the group when they join or leave from the multicast group. Moreover, they do not provide necessary solutions for changing the GK when the group membership changes which is not providing forward/backward secrecy. When a new member joins into the service, it is the responsibility of the KS to prevent new members from having access to previous data. This provides backward secrecy, in a secure multimedia communication. Similarly, when an existing group member leaves from any group, he or she should not have further access to the multicast communication which provides forward secrecy. The backward and forward secrecy can be achieved only through the use of dynamic GK management schemes. In order to provide forward and backward secrecy, the keys are frequently updated whenever a member joins or leaves the multicast service. Furthermore, if a device lacks storage capabilities, it may be impossible within the receiving device to implement a GK management protocol based on a key tree or key-star structure. Hence, the amount of information to be stored to find the updated key by KS and group members should also be minimised. KS also takes care of the job of distributing the secret key (private key) and GK in a secure way using secure socket layer or by using smart cards to the group members.

In this paper, we propose a new centralised GK management scheme based on the Chinese remainder theorem (CRT) to update and distribute the GK with less computational complexity to the multicast group members. In our proposed algorithm, the KS performs only one addition operation for updating the GK when a new user is joined in the dynamic multicast group. Similarly, it performs only one subtraction operation when an existing user is left from the multicast group. Moreover, each member of the multicast group is allowed to perform only one modulo division operation for recovering the updated key when there is a change in the multicast group. Therefore, our proposed CRT-based GK management (CRTGKM) algorithm takes less amount of computation complexity by slightly increasing the storage space of the KS. Moreover, the proposed algorithm also makes group members to store only two key values for receiving the multicast data. Although our proposed algorithm takes less computation power both in KS and group members side, it does not need to maintain any complex hierarchical key structure.

The remainder of this paper is organised as follows: Section 2 provides the features of some of the related works. Section 3 discusses the proposed GK management algorithm and a detailed explanation of the proposed work. Section 4 explains the integration of our proposed approach in a key-star-based key management approach. Section 5 analyses the performance of our proposed algorithm with the other existing centralised key management schemes. Section 6 gives concluding remarks and suggests some future directions.

2 Literature survey

There are many works that are present in the literature on centralised GK management schemes [1–11]. In most of the existing centralised key management schemes, different types of group users obtain a new distributed multicast GK which is used for encrypting and decrypting multimedia data for every session update. However, most of these schemes consume more key computation time and memory, and in addition most existing schemes take more broadcast messages.

Wallner *et al.* [3] discussed about the difficult problem of key management for multicast communication sessions. It focuses on two main areas of concern with respect to key management, which are initialising the multicast group with a common GK and rekeying the multicast group. Even though, these authors achieved efficiency in terms of rekeying cost and memory, it must be further enhanced to provide better performance by avoiding delays in packet transmission. Steiner *et al.* [12] proposed a new key management protocol based on the Diffie–Hellman key exchange. This protocol achieves secure and efficient key agreement in the context of dynamic peer groups which are relatively small and non-hierarchical. Their protocol is efficient only for small groups and not suitable for large groups. Wong *et al.* [13] presented a novel solution to the scalability problem of group or multicast key management. They introduced the concept of key graphs for specifying secure groups. In addition, they presented three strategies for securely distributing rekey messages after a join and leave and proposed new protocols for joining and leaving a secure group. The rekeying strategies and join and leave protocols have been implemented in a prototype KS that they have built. The main limitation of this approach has increased computation complexity.

Trappe *et al.* [14] controlled the problem of access to multimedia multicasting which requires that it is necessary to have the key distribution and maintenance information. The conventional approach for distributing keys is to use a channel independent of the multimedia content. They proposed a second approach, which involves the use of a data-dependent channel, and achieved for multimedia by using data embedding techniques. Poovendran and Baras [15] showed that the rooted-tree-based secure multicast key distribution schemes can be useful for collision avoidance and reduces memory requirements. Li *et al.* [16] studied the problem of distributing cryptographic keys to a secure multicast group with a single sender and multiple receivers. They showed that the problem of designing a key distribution model with specific communication overhead can be posed as a constraint optimisation problem. Using the formulation, they showed how to minimise the number of keys to be stored by the group controller. The main advantage of their work is that they provided security for one to many communications. However, the solution to the constraint optimisation problem itself is much more complex.

Trappe *et al.* [17] presented two modes of conveyance for transmitting the rekeying messages. By embedding the keying information in the multimedia content, the key updating messages associated with secure multicast key management schemes have been hidden in the data in their work and they used it in conjunction with encryption to protect the data from unauthorised access. However, embedding key is also needed for providing media depending key distribution and hence increases more computation time. David *et al.* [18] presented and analysed a new practical centralised hierarchical algorithm for establishing shared cryptographic keys for large, dynamically changing groups. Unlike previously proposed solutions based on information theory and hybrid approaches, the one-way function algorithm proposed by them has communication, computation and storage requirements that scale logarithmically with group size, for adding or evicting operation.

The problem of designing a storage efficient secure multicast key management scheme based on one-way function trees for a pre-specified key update communication overhead was proposed by Li *et al.* [19]. Wang *et al.* [20] proposed an efficient time-bound scheme based on a technique called merging. The idea behind merging is to consider primitive keys instead of hierarchies. It is conceptually like the compression used in source coding. Through this technique, it is feasible to combine multiple keys into an aggregate key. Thus, communication and storage requirements are greatly reduced. Je *et al.* [4] proposed a computation-and-storage-efficient key tree structure, and a key tree management protocol for secure multicast communication. By considering the resource information of each group member's device, this protocol manages the key tree structure to maximise the efficiency of the computation and storage costs and minimises the increment of the communication cost.

Zheng *et al.* [5] proposed two centralised GK management protocols based on the CRT (Chinese remainder GK (CRGK) and fast Chinese remainder GK (FCRGK)). The main advantage of their approach is that the number of broadcast messages to distribute the GK to user side is minimised. In addition, user side key computation is minimised. However, the main limitation of their approach is that computation complexity of KS is very high. Zhou and Ou [6] proposed CRT-based static key structure for distributing the GK to the members of the group when group membership changes. The main contribution of this work is that root ID construction algorithm by using linked list data structure which minimises broadcast messages required to distribute the GK to group members. It also minimises user side key computation. However, it also increases the workload of KS by allowing the KS to find a common GK by using CRT for '*n*' number of congruential equations. Xu and Huang [21] proposed a new multicast key distribution scheme in which the computation complexity is reduced by using maximum distance separable codes to distribute multicast key dynamically. Naranjo *et al.* [22] presented a new algorithm for key management in which they explained three applications of their algorithm to security and privacy field. The main limitations of these existing works are the computation complexity involved in rekeying operations leading to decrease in performance. In addition, the memory requirements are high in most existing schemes. Vijayakumar *et al.* [1] proposed a greatest common divisor-based key distribution protocol that focuses on two dimensions. The first dimension deals with the reduction of computation complexity and second dimension aims at reducing the amount of information stored in the GC and group members while performing the update operation in the key content. The computation cost of GC is $O(3+G)$ and the user has to perform 1 mod, 1 multiplication and 1 multiplicative inverse operations. Moreover, the storage complexity of GC is $((n \times k)+2k+1)$ where k is the number of clusters and user's storage complexity is three keys. Comparing with all these existing schemes, the key management proposed in this paper are more efficient in terms of computation, communication and storage aspects.

When the number of joining or leaving operations is more, batch joins and leaving operations has been proposed in the past [7, 8, 23–28] to perform batch joining and leaving operation. Steve Li *et al.* [23] presented a new technique for multicast batch rekeying. This technique reallocates the tree nodes in order to keep the tree balanced all the time. Perrig *et al.* [24] introduced efficient large-group key distribution, an efficient, scalable and secure method for distributing GKs. This technique has widespread applications, such as access control in streaming multimedia broadcasts. Brian Zhang *et al.* [25] presented the design and implementation of a new key management protocol that is scalable and reliable with respect to performance. The protocol is based on the use of key trees for secure groups and periodic batch rekeying. Goshi and Ladner [26] proposed a height-balanced 2–3 tree (B-tree of order $m=3$) and found that it has the best performance among the balancing strategies tested. However, balancing a B-tree after member joining involves splitting oversised tree nodes and results in worst-case rekeying cost. Lu [27] developed a non-split balancing high-order (NSBHO) tree. Unlike the B-tree scheme, their NSBHO tree does not use node splitting to balance the tree. Their experiments confirmed that the NSBHO-tree is superior to the B-tree in terms of the worst-case rekeying performance. In addition, it has better average-case rekeying performance. Hock Desmond Ng *et al.* [28] presented two merging algorithms that are suitable for batch joining events. To additionally handle batch depart requests, they have extended these two merging algorithms into a batch balanced algorithm. All these algorithms require the GC to update the affected members on their node position by using update messages. By minimising the differences in height, they minimised the number of key storages and decryptions needed by each member. This is critical for terminals with limited

Table 1 Overall summary of literature survey

Name of the system	Year	Merits	Demerits
Wallner *et al.* [13]	1998	✓ Proposed multicast key management to provide secure group communication	✓ Computation complexity is high
Steiner *et al.* [12]	2000		
Trappe *et al.* [14]	2001		✓ Storage complexity is high
Poovendran and Baras [15]	2001	✓ Scalability is increased	
Je *et al.* [4]	2010		
Trappe *et al.* [17]	2003	✓ Proposed a media dependant channel-based key distribution	
David *et al.* [18]	2003	✓ Proposed a key management in which computation and communication complexity is increased logarithmically with group size	
Li *et al.* [16]	2002	✓ Proposed a Storage efficient key management scheme in which storage and communication complexity is minimized	✓ KS and group members computation complexity is high
Li *et al.* [19]	2004		
Wang and Laih [20]	2006		
Zheng *et al.* [5]	2007	✓ Proposed a CRT-based key management to provide high security with less storage and communication complexity	
Jie Zhou *et al.* [6]	2009		
Naranjo *et al.* [22]	2012	✓ Proposed an authenticated key management algorithm using extended Euclidean algorithm	
Vijayakumar *et al.*	article online	✓ Proposed a clustered key tree management scheme	
Steve Li *et al.* [23]	2001	✓ Proposed key management techniques for multicast batch rekeying operations	✓ Rekeying cost is high for batch joining and leaving operations ✓ Communication complexity is high
Perrig *et al.* [24]	2001		
Brian Zhang [25]	2003		
Goshi *et al.* [26]	2003		
Lu *et al.* [27]	2005	✓ Proposed various storage efficient key management techniques such as 2–3 tree, merging, batch balanced, *m*-dimensional space geometry and tree-based algorithms to support batch rekeying operations	✓ It is not suitable for the case where batch leaves are higher than batch joined ✓ Communication complexity is also high
Hock Desmond Ng *et al.* [28]	2007		
Xie *et al.* [7]	2009		
Bezawada *et al.* [8]	2011		

computation and storage. Furthermore, reducing the number of decryptions helps to reduce the energy consumption, which in turn leads to battery saving.

Xie *et al.* [7] proposed an *m*-dimensional space geometry sphere rekeying scheme, in which GC generates parent's key by its child's key when member join or leave, so the communication cost of GC for rekeying is reduced. When many of members join or leave at the same time, their batch rekeying scheme improves the rekeying performance. By their simulation, they have shown that their scheme decreases communication cost and storage cost, increase new GK distribution efficiency and improves expansibility of multicast rekeying. Bruhadeshwar *et al.* [8] presented a family of algorithms for effective key management that reduces the number of keys that are used by the users and the time required for rekeying because of the revocation of multiple users. They showed that their algorithms reduce the cost of rekeying by 43–79% when compared with the previous solutions. Table 1 shows the overall summary of the centralised GK management approaches.

Comparing with most of the existing key management schemes that are existing in the literature, the key management scheme proposed in this paper is different in many ways. First, the computation complexity of KS and group user is reduced substantially by minimising the number of arithmetic operations taken by GC and group user. In order to minimise the computation time both in KS and user side, we use CRT-based key management scheme. The motivation for using CRT in our work is that there are many multimedia multicast applications in which more than one user is participating in order to securely access the multimedia data transmitted from a trusted third party. In such a scenario, the GK used for protecting the multimedia data can be computed from a system of congruences established by CRT that contains each user's secret key. Moreover, CRT-based GK management approach gives high security against various attacks.

Second, comparing with all the existing batch rekeying algorithms, the batch rekeying or batch updating algorithm proposed in this paper is more efficient in terms of computation power taken by the GC and the number of key values stored by group members are also minimised. Finally, the proposed algorithm also reduces the amount of information that needs to be communicated for updating the keys when there is a change in the group membership.

3 Proposed CRTGKM algorithm

In this section, we introduce our proposed CRTGKM algorithm which is based on the CRT. This section is divided into two subsections. The first Section 3.1 gives an overview of CRT and second Section 3.2 explains about our proposed CRTGKM algorithm which is based on CRT.

3.1 Chinese remainder theorem

Let $k_1, k_2, k_3, \ldots, k_n$ be pairwise relative prime positive numbers, and let $a_1, a_2, a_3, \ldots, a_n$ be positive integers. Then, CRT states that the pair of congruences

$$\begin{gathered} X \equiv a_1 \bmod k_1 \\ X \equiv a_2 \bmod k_2 \\ \vdots \\ X \equiv a_n \bmod k_n \end{gathered}$$

has a unique solution mod $\partial_g = \prod_{i=1}^{n}(k_i)$. To compute the unique solution, the KS can compute the value as shown in (1)

$$X = \sum_{i=1}^{n} a_i\beta_i\gamma_i\left(\bmod\ \partial_g\right)$$
$$\text{where } \beta_i = \frac{\partial_g}{k_i} \quad \text{and} \quad \beta_i\gamma_i \equiv 1 \bmod k_i \tag{1}$$

3.2 CRTGKM algorithm

The proposed CRTGKM algorithm works in three phases. The first phase is the KS initial set up, where a multiplicative group is created at KS. In the second phase called the user initial join phase, the users are sending join requests to the KS and obtain all the necessary keys for participation through secure channel. The final phase of this protocol is known as user leaves phase that deals with updating of GK when a user is left from the dynamic multicast group in order to provide forward secrecy. The proposed work mainly concentrates on the third phase called 'key updation' phase because the computation time is extremely large in most of the existing GK management systems. This is extremely a great challenge in most of the multimedia multicast applications.

3.2.1 KS initial set up: Initially, the KS selects large prime numbers p and q, where $p > q$ and $q \le \lceil p/4 \rceil$. The value p helps for defining a multiplicative group z_p^* and q is used to fix a threshold value to select the GK values. Initially, the KS selects secret keys or private keys k_i from the multiplicative group z_p^* for 'n' number of users which will be given to users as they join into the multicast group. In the CRT-based scheme, we require that all the private keys selected from z_p^* are pairwise relatively prime positive integers as explained in [5, 6, 22]. Moreover, all the private keys should be much larger than the GK which is selected within the threshold value fixed by q. Next, KS executes the following steps in the KS initialisation phase.

(1) Compute $\partial_g = \prod_{i=1}^{n}(k_i)$ as we used in our previous approach [1].
(2) Compute

$$x_i = \frac{\partial_g}{k_i} \quad \text{where } i = 1, 2, 3, \ldots, n \tag{2}$$

(3) Compute y_i such that

$$x_i \times y_i \equiv 1 \bmod k_i \tag{3}$$

(4) Multiply all users x_i and y_i values and store them in the variables

$$\text{var}_i = x_i \times y_i \tag{4}$$

(5) Compute the value

$$\mu = \sum_{i}^{n} \text{var}_i \tag{5}$$

3.2.2 User initial join: Whenever a new user 'i' is authorised to join the dynamic multicast group for the first time, the KS sends a secret key k_i using a secure unicast which is known only to the user u_i and KS. Next, KS computes the GK in the following way and broadcast it to the users of the multicast group.

(a) Initially, KS selects a random element k_g as a new GK within the range q.
(b) Multiply the newly generated GK with the value μ (computed in KS initial set up phase) as shown in (6)

$$\gamma = k_g \times \mu \tag{6}$$

(c) The KS broadcast a single message γ to the multicast group members. On receiving γ value from the KS, an authorised user u_i of the current group can obtain the new GK k_g by doing only one mod operation as shown in (7)

$$\gamma \bmod k_i = k_g \tag{7}$$

The k_g obtained in this way must be equal to the k_g generated in step (a) of user initial join phase. When 'i' reaches to n, KS executes KS initial set up phase to compute ∂_g, var_i and μ for 'm' number of users where $m = n \times \delta$. The value δ is a constant value which may take values < 5 depending on the dynamic nature of the multicast group.

3.2.3 User leave: Group key updating operation when a user leaves usually takes more computation time in most of the GK management protocol since the KS cannot use the old GK to encrypt the new GK value. When a new user joins the service, it is easy to communicate the new GK with the help of the old GK. Since the old GK is not known to the new user, the newly joining user cannot view the past communications. This provides backward secrecy. User leave operation is completely different from user join operation. In user leave operation, when a user leaves the group, the KS must avoid the use of an old GK to encrypt a new GK. Since the old user knows the old GK, it is necessary to use each user's secret key to perform a rekeying operation when a user departs from the services. In our proposed key management scheme, the GK updating processes when a user leave operation is performed in a simplest way. When a user u_i leaves from the multicast group, KS has to perform the following steps:

(1) Subtract var_i from μ

$$\mu' = \mu - \text{var}_i \tag{8}$$

(2) Next, KS must select a new GK and it should be multiplied by μ' to form the rekeying message as shown below

$$\gamma' = k_g' \times \mu' \tag{9}$$

(3) The updated GK value will be sent as a broadcast message to all the existing group members. The existing members of the multicast group can obtain the updated GK value k_g' by doing only one mod operation as shown in (7). From the received value, the user u_i cannot find the newly updated GK k_g' since his or her component is not included in μ'.

4 CRTGKM on key-star-based approach

The proposed CRTGKM approach can be integrated into a key-star-based approach for minimising user storage and to update the GK with minimal computation complexity. Fig. 1 shows a key-star in which the root is the GK, leaf nodes are the individual user's secret keys. The key-star shown in Fig. 1 consists of only two levels. The first level (0th level) consists of the GK. The second level (1st level) contains the secret keys, k_i where $i = 1, 2, \ldots, n$ which are known only to the KS and group member. Each user is attached to the second level and they are given with two keys namely secret key and GK. In this key-star-based GK management scheme, updating is necessary for each rekeying operation used for user leave and user join operations. For example, if a user u_3 from Fig. 1 leaves the group, then the GK k_g must be changed and the updated GK k_g' is sent as a broadcast message to the remaining group members.

In order to update the GK during member leave operation, the KS can use the steps defined in Section 3.2.3. Therefore, if a user leaves from the dynamic multicast group, only one subtraction operation is to be performed to update the GK.

4.1 Batch leave

If a batch of users wants to leave from the multicast group, then the KS has to perform more than one addition operation for updating the GK. For example, if the users u_3, u_5, u_7 and u_9 want to leave from the multicast group, then KS has to perform the following steps for GK updating.

(1) Subtract var_3, var_5, var_7 and var_9 from μ. That is, $\mu' = \mu - (\text{var}_3 + \text{var}_5 + \text{var}_7 + \text{var}_9)$.
(2) Next, KS selects a new GK k_g' and multiplies it with updated μ' to form the rekeying message as shown in (9).
(3) The updated GK value will be sent as a broadcast message to all the existing group members. The existing members of the multicast group can obtain the updated GK value k_g' by doing only one mod operation as shown in (7). From the broadcast value γ', the users u_3, u_5, u_7 and u_9 cannot find the newly updated GK k_g' since their components are not included in μ'.

Therefore, in general, if 'n' users want to leave from the dynamic multicast group, the KS has to perform $(n - 1)$ additions + 1 subtraction for updating the GK.

4.2 Batch join

If a batch of users wants to join into the multicast group, then the KS has to perform more than one addition operation for updating the GK. For example, if four users u_3, u_5, u_7 and u_9 want to join into already existing multicast group, then KS has to perform the following steps for GK updation.

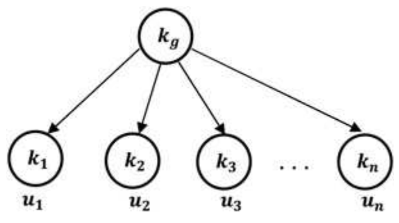

Fig. 1 *Key-star-based group key management*

(1) Instead of computing x_i and y_i for all the four users, KS can take the multiplied values of x_i and y_i from the variables var$_3$, var$_5$, var$_7$ and var$_9$ which are already computed in the KS initialisation phase. KS can select those values from the GC storage area to compute $\mu' = \mu + (\text{var}_3 + \text{var}_5 + \text{var}_7 + \text{var}_9)$.
(2) Next, KS selects a new GK k'_g and multiplies it with updated μ' to form the rekeying message as shown in (9).
(3) The updated GK value will be sent as a broadcast message to all the existing and newly joined group users. From the broadcast value γ', all the users of the multicast group can find the newly updated GK k'_g since their var$_i$ values are included in μ' by using var$_3$, var$_5$, var$_7$ and var$_9$.

Therefore, in general, if 'n' users want to join into the existing dynamic multicast group, the KS has to perform (n) additions for updating the GK. The key strength of our algorithm is that computation complexity of KS is completely reduced in comparison with the other existing approaches [5, 6]. The computation complexity of a KS is $O(1)$ when a single user joins or leaves from the multicast group. In addition to this, the computation complexity of a multicast group user is also minimised by allowing each user to perform only one modulo division operation. Moreover, KS takes only one broadcast message which is same in most of the existing algorithms for informing the updated GK value to users of the multicast group. Table 2 shows the computation and storage complexities of various key management approaches CRGK, FCRGK [5] and key tree CRT (KTCRT) [6] which are based on a CRT. The notations used for comparisons are defined as: n is the number of users, τ is the maximum number of children of each node of the tree, EEA is the time taken to find the inverse element of a multiplicative group using extended Euclidean algorithm, M is representing multiplication operation, A is addition operation and S is representing the subtraction operation.

4.3 Security analysis

This paper analyses the proposed algorithm for forward secrecy and backward secrecy. The assumption of the implemented protocol is that an adversary might be a group member sometime and KS keeps all users private key as secret and each user will keep their private as secret.

4.3.1 Backward secrecy: 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 GK. Moreover, if the adversary becomes a group member, it may try to derive the previous GK which is not permitted. In the proposed protocol, when the newly updated GK is communicated to old group members, an adversary needs to find any one of the group users private key. Moreover, the private keys are randomly selected from a large set of positive integers with respect to the multiplicative group. This property makes the situation infeasible for the adversary to compute any other users secret key. Consequently, the adversary cannot access the communication sent before join, which means the proposed approach supports the initial security requirement.

4.3.2 Forward secrecy: Forward secrecy is the technique of preventing an old member from accessing current communication after leave operation. When a member leaves the group, he or she may try to derive the GK by using any attacking methods. In the proposed algorithm, it is infeasible for an adversary to compute the current GK after the leave for the same reasons that was explained for the backward secrecy technique. Since when a user u_i leaves from the group, KS subtracts his or her share value such as multiplication of x_i and y_i which is stored in var$_i$ from μ value to produce μ'. This updated μ' is multiplied by the newly generated GK value k'_g to form the rekeying message γ'. Therefore a user who had already left from the service cannot find the new GK in a feasible way since his or her personal keying information is not included. The user who had left from the group may try to find k'_g from the rekeying value which is sent as a broadcast message from the KS in an infeasible method. In order to do that the user has to multiply his or her private key value with all the numbers starting from 1 to q where q is the maximum limit of GK value. At a certain point, it will give a value $\vartheta = \gamma'$ (i.e. $k_i \times \omega = \vartheta$). After finding this ω value, the user u_i can find a set of numbers S that will divide the number ω. Therefore, the value of S is defined as the set of numbers $\{\omega \bmod 1, \omega \bmod 2, \ldots, \omega \bmod \omega\} = 0$. Among the set of numbers, newly generated GK k'_g is also one of the number (i.e. $k'_g \in S$). In this case, if the size of k_i is w bits, then the attacker has to perform 2^w multiplication. The time taken to derive k'_g 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 1024 bits and prior experiments were conducted with 128, 256 and 512 bits. After finding the set of values S that divides the number ω, the attacker (user left from the group) can find the new GK by selecting the values from the set S by using brute force attack. If the time required to perform a single attempt using brute force attack is 1 μs, then the total time required will be 2^{s-1} μs. Consequently, an adversary cannot find the GK in order to access the current communication, which means the second security requirement is also supported in our proposed algorithm.

4.3.3 Collusion attack: Collusion attack is the one in which two or more adversaries act as legitimate members when they are participating in the group and then cooperatively compute the updated GK after leaving the group. Since the value of var$_i$ is subtracted from μ after the leaving operation is performed in a multicast group, any number of previous users collision will not be used to gain information about the congruence system and to derive the

Table 2 Computation, storage and communication complexities of various group key management schemes

Parameters	CRGK	FCRGK	KTCRT	CRTGKM (proposed method)
computation cost (KS)	$O(n)(xor + A + M + \text{EEA})$	$O(n)(xor + A + M)$	$O(\log_\tau n)(xor + A + M + \text{EEA})$	$O(1)(A \text{ or } S)$
computation cost (user)	1 mod + 1 *xor*	1 mod + 1 *xor*	1 mod + 1 *xor*	1 mod
storage complexity (user)	2	2	$(\log_\tau n)$	2
storage complexity (KS)	$2n+1$	$4n+1$	$2n-1$	$4n+3$
communication complexity	1 broadcast	1 broadcast	1 broadcast	1 broadcast

updated GK k'_g as long as the pairwise relatively prime numbers are large. The following scenario describes a kind of collusion attack in which two adversaries act as legitimate members. In Fig. 1, suppose that u_1 is an adversary A who knows the key values k_1, k_g and u_3 is an adversary B who knows the key values k_3 and k_g at time $(t-2)$. In time $(t-1)$, the adversary A leaves the group with the key values k_1 and k_g. B receives the rekeying message γ' from the KS at the time (t) and computes k'_g. In time $(t+1)$, B leaves the group with the two key values k_3 and k'_g. Both of these adversaries exchange their known key values k_1, k_g, k_3 and k'_g. Using these known values, the adversaries A and B cannot cooperatively find the updated GK $(k'_g)'$ which is broadcast at time $(t+2)$ in a feasible amount of time since their shares var_1 and var_3 are excluded from μ.

5 Performance analysis

The proposed method has been implemented in JAVA (Intel Core i3 processor, 2GB random access memory, 500 GB hard disk, Windows XP operating system) for a group of 1000 users and we have compared the computation time for various key sizes with the existing approaches to perform the rekeying operation. For implementing this proposed approach, a key-star is created for which 1000 private key values are generated randomly and assigned to the second level of key-star. The private keys used in our approach are 1024 bit positive integers which are relatively prime. For generating large integers in our program, we use Biginteger class that supports various methods for handling large positive integers. The method multiply() supported by Biginteger class is used to multiply all users secret key into a variable which will be used to find x_i and y_i values. The method modInverse() is used to find multiplicative inverse of a given element with respect to the size of the multiplicative group. Table 3 compares the measured computation time which is defined as the time taken to compute the GK at the KS area and key recovery time at group users area for various key management approaches.

For computing the GK in three different existing algorithms present in the literature, we measured the computation time separately for x_i which is obtained by dividing ∂_g and y_i which is obtained by finding the multiplicative inverse for x_i. All the existing algorithms shown in Table 2 take much more computation time for calculating x_i and y_i values which would increase the computing load of the KS and hence may not be desirable for many wired (pay-TV systems and video conferences) and wireless applications where portable devices or sensors need to reduce their computation as much as possible because of battery power limitations. Moreover, for comparing the computation time of the proposed algorithm in milliseconds (ms) with existing algorithms, the key size is taken to be 64, 128, 256, 512 and 1024 bits. It is evident from the values that the GK computation time and key recovery time in our proposed algorithm is found to be better both in the KS area and the client area than the other algorithms. It is to be noted that in our algorithm the key computation time taken by single user join or leave is completely minimised when compared with other existing algorithms for 'n' users.

The reasons behind this reduction in computation time are (i) calculating x_i and y_i value is neglected by storing them in KS storage area and (ii) mutliplying x_i with y_i is also reduced

Table 3 Group key computation and recovery time for various key management approaches

Key size	CRGK				FCRGK				KTCRT				CRTGKM (proposed method)			
	KS, ms			User, ms	KS, ms			User, ms	KS, ms			User, ms	KS, ms			User, ms
	x_i	y_i	Key updation	Key recovery	x_i	y_i	Key updation	Key recovery	x_i	y_i	Key updation	Key recovery	x_i	y_i	Key updation	Key recovery
64 bits	218	203	425	0.4	200	192	402	0.39	202	189	396	0.4	0	0	4	0.23
128 bits	577	562	1148	0.9	462	408	896	0.8	402	380	786	0.8	0	0	7	0.53
256 bits	1671	1780	3551	3.2	1362	1289	2796	2.8	1302	800	2172	2.1	0	0	13	1.5
512 bits	6611	7111	13869	6	4610	4820	9972	6.2	3372	3003	6492	6.2	0	0	19	5.3
1024 bits	25 767	25 913	51 880	12	22 638	20 219	42 857	10.9	18 978	16 254	35 232	11.6	0	0	78	8.4

which is done in the KS initialisation phase. Therefore, our proposed CRTGKM approach reduces the computing load of KS by slightly increasing the storage overhead of the KS. This could be compromised by using a fast and efficient server with peta bytes of storage area. Moreover, the amount of keying information to be stored by the KS is directly proportional to the number of users in our approach. The tree-based schemes must maintain a set of keys for the intermediate logical nodes of the tree and subsets of those keys must be held by the consequent users which would increase user storage space. The storage cost of KS in our proposed algorithm is $4n+3$, where $4n$ represents four types of values (users secret key, x_i, y_i and var_i) to be stored for 'n' numbers of users. The value 3 represents three permanent storage locations used for storing ∂_g, μ and γ values.

The graphical result shown in Fig. 2 is used to compare the KS key computation time of the proposed method with the existing methods. It compares the results obtained from our proposed CRTGKM with CRGK, FCRGK and KTCRT. From Fig. 2, it is observed that when the key is 512 bits, the GK computation time is found to be 19 ms in our proposed approach, which is better in comparison with the other existing schemes. Moreover, if the number of members who are joining and leaving increases, then the computation time is around 50 ms in our proposed approach which is very small in comparison with existing schemes and it is not shown in Fig. 2. The result shown in Fig. 3 is used to compare the user's key recovery time of our proposed method with the existing methods. It compares the results obtained from our proposed scheme with existing approaches and it is observed that when the key size is 512 bits, the key recovery time of a user is found to be 5.3 ms in our proposed approach, which is better in comparison with the other existing schemes.

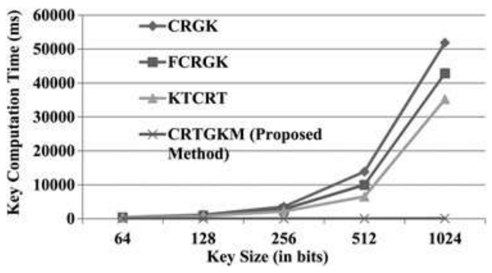

Fig. 2 *Group key computation time at KS*

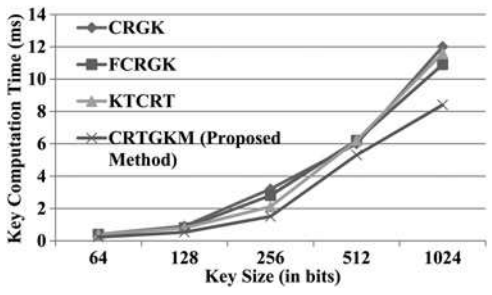

Fig. 3 *User key recovery time for obtaining the new group key value*

6 Concluding remarks

This paper proposes a new solution to reduce the computation complexity without increasing the storage complexity for providing secure multimedia multicast through effective GK management technique which is based on CRT. For this purpose, a new key-star-based key management protocol (key value = 'n' bit numbers) has been proposed in this paper. The proposed algorithm focuses mainly on the minimising KS and user's computation complexity. When the key size is small (key size = 128 bits), the computation time decreases by around 700 ms in the KS area and when the key size increases (1024 bits) the computation time decreases by around 35 000 ms for updating a single key from the dynamic multicast group. With regard to the storage complexity, the amounts of keys stored by group members are almost same and a slight increase in KS storage space. Moreover, our proposed algorithm also takes single broadcast message for informing the group members in order to recover the updated keying information thereby maintaining the same rekeying cost both for batch join and batch leave operations.

7 References

1 VijayaKumar, P., Bose, S., Kannan, A.: 'Centralized key distribution protocol using the greatest common divisor method', *Comput. Math. Appl.*, **65**, (9), pp. 1360–1368 doi:10.1016/j.camwa.2012.01.038

2 VijayaKumar, P., Bose, S., Kannan, A.: 'Rotation based secure multicast key management for batch rekeying operations', *Netw. Sci.*, 2012, **1**, (1–4), pp. 39–47

3 Wallner, D.M., Harder, E.J., Agee, R.C.: 'Key management for multicast: issues and architectures', Internet Draft Report, Filename: draft-wallner-key-arch-01.txt, 1998

4 Je, D.-H., Lee, J.-S., Park, Y., Seo, S.-W.: 'Computation-and-storage efficient key tree management protocol for secure multicast communications', *Comput. Commun.*, 2010, **33**, (6), pp. 136–148

5 Zheng, X.L., Huang, C.T., Matthews, M.: 'Chinese remainder theorem based group key management'. Association for Computing Machinery Proc. 45th Annual Southeast regional Conf. (ACMSE-07), Winston-Salem, North Carolina, USA, 2007, pp. 266–271

6 Zhou, J., Ou, Y.-H.: 'Key tree and chinese remainder theorem based group key distribution scheme', *J. Chin. Inst. Eng.*, 2009, **32**, (7), pp. 967–974

7 Xie, H.-T., Yang, Z.-K., Wang, Y.-M., Cheng, W.-Q.: 'A m-dimensional sphere multicast rekeying scheme', *Commun. Mob. Comput. Int. Conf.*, 2009, **3**, pp. 418–422

8 Bruhadeshwar, B., Sandeep, S., Kulkarni: 'Balancing revocation and storage trade-offs in secure group communication', *IEEE Trans. Dependable Secur. Comput.*, 2011, **8**, (1), pp. 58–73

9 Sakarindr, P., Ansari, N.: 'Survey of security services on group communications', *IET Inf. Secur.*, 2010, **4**, (4), pp. 258–272

10 Wang, H., Qin, B.: 'Improved one-to-many authentication scheme for access control in pay-TV systems', *IET Inf. Secur.*, 2012, **6**, (4), pp. 281–290

11 Chou, Y.H., Chen, C.Y., Chao, H.C., Park, J.H., Fan, R.K.: 'Quantum entanglement and non-locality based secure computation for future communication', *IET Inf. Secur.*, 2011, **5**, (1), pp. 69–79

12 Steiner, M., Tsudik, G., Waidner, M.: 'Key agreement in dynamic peer groups', *IEEE Trans. Parallel Distrib. Syst.*, 2000, **11**, (8), pp. 769–980

13 Wong, C., Gouda, M., Lam, S.: 'Secure group communications using key graphs', *IEEE/ACM Trans. Netw.*, 2000, **8**, (1), pp. 16–30

14 Trappe, W., Song, J., Poovendran, R., Liu, K.J.R.: 'Key distribution for secure multimedia multicasts via data embedding', *IEEE Int. Conf. Acoust. Speech Signal Process.*, 2001, **3**, pp. 1449–1452

15 Poovendran, R., Baras, J.S.: 'An information-theoretic approach for design and analysis of rooted-tree-based multicast key management schemes', *IEEE Trans. Inf. Theory*, 2001, **47**, (7), pp. 2824–2834

16 Li, M., Poovendran, R., Berenstein, C.: 'Design of secure multicast key management schemes with communication budget constraint', *IEEE Commun. Lett.*, 2002, **6**, (3), pp. 108–110

17 Trappe, W., Song, J., Poovendran, R., Ray Liu, K.J.R.: 'Key management and distribution for secure multimedia multicast', *IEEE Trans. Multimed.*, 2003, **5**, (4), pp. 544–557

18 McGrew, D.A., Sherman, A.T.: ‘Key establishment in large dynamic groups using one-way function trees’, *IEEE Trans. Softw. Eng.*, 2003, **29**, (5), pp. 444–458

19 Li, M., Poovendran, R., McGrew, D.A.: ‘Minimizing center key storage in hybrid one-way function based group key management with communication constraints’, *Inf. Process. Lett.*, 2004, **93**, (4), pp. 191–198

20 Wang, W.Y., Laih, C.S.: ‘Merging: an efficient solution for a timebound hierarchical key assignment scheme’, *IEEE Trans. Dependable Secur. Comput.*, 2006, **3**, (1), pp. 91–100

21 Xu, L., Huang, C.: ‘Computation-efficient multicast key distribution’, *IEEE Trans. Parallel Distrib. Syst.*, 2008, **19**, (5), pp. 1–10

22 Naranjo, J.A.M., Lopez-Ramosz, J.A., Casado, L.G.: ‘A suite of algorithms for key distribution and authentication in centralized secure multicast environments’, *J. Comput. Appl. Math.*, 2012, **236**, (12), pp. 3042–3051

23 Steve Li, X., Richard Yang, Y., Gouda, M., Lam, S.: ‘Batch rekeying for secure group communications’. Proc. Tenth Int. Conf. WWW, 2001, pp. 525–534

24 Perrig, A., Song, D., Tygar, J.D.: ‘ELK a new protocol for efficient large-group key distribution’. Proc. IEEE Symp. Security and Privacy Symp., 2001, pp. 247–262

25 Brian Zhang, X., Lam, S., Young Lee, D., Richard Yang, Y.: ‘Protocol design for scalable and reliable group rekeying’, *IEEE/ACM Trans. Netw.*, 2003, **11**, (6), pp. 908–922

26 Goshi, J., Ladner, R.E.: ‘Algorithms for dynamic multicast key distribution trees’. Proc. ACM Symp. Principles of Distributed Computing, 2003, pp. 243–251

27 Lu, H.: ‘A novel high-order tree for secure multicast key management’, *IEEE Trans. Comput.*, 2005, **54**, (2), pp. 214–224

28 Hock Desmond Ng, W., Howarth, M., Sun, Z., Cruickshank, H.: ‘Dynamic balanced key tree management for secure multicast communications’, *IEEE Trans. Comput.*, 2007, **56**, (50), pp. 590–605