International Journal of Computer Network and Security(IJCNS)
Vol 4. No 1. Jan-Mar 2012 ISSN: 0975-8283
www.ijcns.com

Enhancement of Hierarchical Key Management Scheme of Secure Multicast Transmission for Overlay Networks

[1]P.G.Kathiravan, [2]C.Rajan, [3]Dr.N.Shanthi,
[1,2,3]*Department of Information Technology, K.S.Rangasamy college of Technology (Autonomous), Anna University, Coimbatore, Tamilnadu, India.*
[1]*ask2kathiravanit@gmail.com*,[2]*rajanksrct@gmail.com*,[3]*shanthimoorthi@yahoo.com*.

Abstract-Group communication security is needed to protect sensitive information. A group key is shared by all users under secure group communication model. The group key is used to encrypt data transmitted to the group. The group membership is dynamic and requires new key for membership changes. Secure multicast transmission schemes are used to transfer data to a set of nodes. Membership in secure multicast groups is dynamic and requires multiple updates in a single time frame. Storage cost and rekeying cost are considered in the secure multicast transmission. The system manages long standing members and shortly lived members separately. Long standing members need to store smaller number of keys than short-lived members. The system manages variable storage levels for key management. Hierarchical key management algorithm is used to manage group key values. The system manages membership addition and removes operations. The key management scheme is not optimized for overlay networks, End node based multicast transmission is not optimized, Traffic overhead is high and Transmission and listen mode energy levels are not managed problems are identified from the existing security models. The secure multicast transmission supports key management under multicast groups. The storage and key revocation operations are managed by the system. The system is enhanced to manage keys under overlay networks. The traffic and energy control mechanisms are integrated with the system. The system is designed to manage storage and key revocation process. Multicast group and key management operations are integrated in the system. The system is enhanced to manage group keys under overlay network environment. The key management messages are controlled in the system.

Keywords: *Multicast, rekeying, Hierarchical key management, revocations, Overlay networks.*

1.INTRODUCTION

Applications such as conferencing, distributed interactive simulations, networked gaming, and news dissemination are group-oriented. In these applications, it is necessary to secure the group communication as the data are sensitive or it requires the users to pay for it. In the algorithms for secure group communication, a group key is shared by all the users. The group key is used to encrypt data transmitted to the group. The group membership is dynamic. When group membership changes, to protect the confidentiality of the current users, a new group key needs to be shared by the users.

The dynamics of the group membership can be handled under two settings. In the first setting, a central group controller manages the group membership and the users do not have the necessity to communicate among themselves. Scenarios like pay TV, news dissemination, stock information, etc., are in this category. In these scenarios, typically, the group size is large and geographically disparate. In the second setting, the group members collaborate to agree upon a common group key [1]. Applications like conferencing and distributed interactive simulation fall under this category. The group sizes in such applications are typically small and justifies the usage of the relatively high end computation required by the group key agreement techniques [2]. In this work, we consider the first setting where a large group of users is managed by a group controller and consider the cost of membership handling in such applications.

When a user is admitted to the group, the group controller changes the group key and securely unicasts it to the joining user. To send the new group key to the current users, the group controller encrypts it with the old group key and multicasts it to them. Thus, the cost of rekeying for the group controller, due to a joining user is small. However, when a user is revoked, i.e., the user leaves or is forcefully removed from the group, the group controller needs to securely unicast the new group key to each of the remaining users. Toward this, the group controller encrypts the new group key with the personal keys of each of the remaining users and unicasts each message to the respective user. The cost of this process is O(N) symmetric key encryptions and O(N) messages. Thus, for a large group, revoking users from the secure group is an expensive operation.

Many solutions have been proposed for efficiently handling a single membership change, i.e., a single join or revocation of a user. In these solutions, for a group of N users, the group controller distributes the new group key in O (logN) encrypted messages. We note that in these solutions, the rekeying cost, i.e., number of encryptions performed and messages transmitted by the group controller, for a joining user is increased from two to O(logN). However, techniques suggested reduce the join cost to nearly constant and as such have been used by other approaches [5], [6]. On the other hand, the cost for revoking a user is reduced from O(N) to O(logN) encrypted messages. However, to handle multiple membership changes, the group controller repeats the process of revocation for each revoked user. Optimizations such as batch or periodic rekeying reduce this cost to some extent. However, even in these solutions, the cost of revocation is high. Moreover, as the group controller needs to interrupt the group communication during the rekeying, the resulting delay can be unreasonable for many applications. Thus, efficient distribution of the new group key for multiple membership changes is a critical problem in secure group communication.

One approach to revoke multiple users is to associate a key with every nonempty subset of users in the group. Thus, if one or more users are revoked, the group controller uses the key associated with the subset of the remaining users to encrypt the new group key and transmits the new group key to them. The advantage of this approach is that the communication overhead is only one message for revoking any number of users.

However, the number of keys stored by the group controller and the users is exponential in the size of the group. In this paper, we describe a family of key management algorithms that reduce the cost due to multiple user revocation while keeping the storage cost manageable. The goal of the paper is to evaluate trade-off between storage and revocation cost. Storage is computed in terms of keys that each user maintains [9]. And revocation cost is computed in terms of the encryptions performed, and the number of messages transmitted, by the group controller. Similar to the algorithms, we assume that the communication from the group controller is broadcast in nature. Using our algorithms, the group controller can efficiently distribute the group key.

Notations. We use k(m) to denote that message m is encrypted with key k. Only users who know k can decrypt this message. The adversary can listen to all messages sent over the network. Hence, for simplicity, we assume that all communication is broadcast in nature, and hence, we do not explicitly identify the intended recipients of a message.

I. RELATED WORK

Other approaches to address the problem of revoking multiple users are proposed in [3]. The group controller maintains a logical hierarchy of keys that are shared by different subsets of the users. To revoke multiple users, the group controller aggregates the entire necessary key updates to be performed and processes them in a single step.

The group controller interrupts the group communication until all the necessary key updates are performed, and then, distributes the new group key to restore group communication. This interruption to group communication is undesirable for real-time and multimedia applications. To handle multiple group membership changes, the group controller performs periodic rekeying, i.e., instead of rekeying whenever group membership changes, the group controller performs rekeying only at the end of selected time intervals. However, the revoked users can access group communication until the group is rekeyed.

This can either cause monetary loss to the service provider or compromise confidentiality of other users. The group controller maintains a logical hierarchy of keys similar to the solution. To revoke multiple users, the group controller distributes the new group key by using keys that are not known to the revoked users.

However, this solution achieves a good rekeying cost only if the size of the revoked users is either very small or very large. In the above schemes, the logical key tree structure tends to become unbalanced after some membership changes and results in tree which has large height (O (N)). As the height of the tree determines the rekeying cost, several approaches [7] have been proposed to address this issue. These approaches focus on algorithms for reorganizing the tree structure that becomes unbalanced after a few membership changes.

However, the basic rekeying algorithm. The approaches in these works are orthogonal to our algorithms in that the approaches from these works can be used to balance the tree used in our algorithms.

The authors describe an information-theoretic approach for analyzing key-tree-based protocols and show interesting relationships among the storage cost, the number of rekeying messages, and the resistance against colluding users. They describe an optimal key distribution protocol which is weakly collusion resistant, i.e., it cannot tolerate collusion of two users.

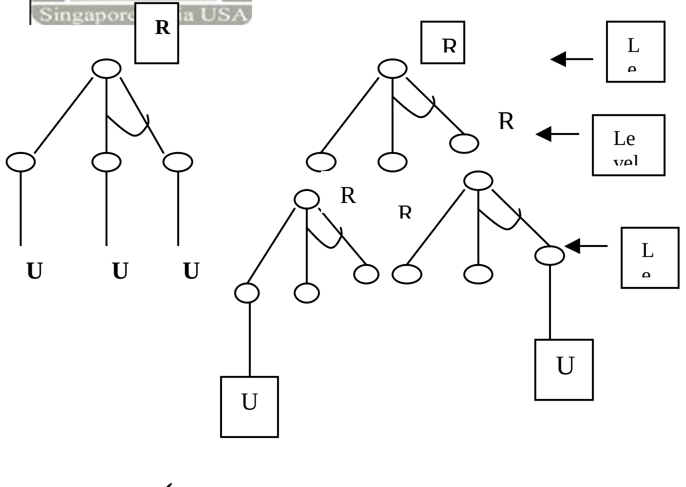

Fig. 1. Partial view of (a) basic structure (b) hierarchical structure.

The authors focus on the storage versus communication trade-offs in secure conferencing given offline and interactive key distribution models. However, their approaches do not address the issue of rekeying as they focus on a fixed-size coalition of attackers and perform an appropriate key distribution to address this threat model. Luby and Staddon focus on the trade-off between the storage cost and the rekeying cost. They identify a lower bound on the rekeying cost based on the number of keys that the users maintain. Their work is based on previous work and assumes that an upper bound on the number of users, say x, that need to be revoked is known in advance. The key distribution algorithm uses the value of x to distribute the keys. Hence, if the number of users that need to be revoked is more than x, then their algorithm fails to revoke them using the shared keys. By contrast, our algorithm does not assume that the number of revoked users is known in advance.

II. KEY MANAGEMENT ALGORITHMS

3.1 The Basic Structure

We arrange a group of K users as children of a rooted tree, as shown in Fig. 1a. Let R be the root node. We use the tuple <R, u_1, u_2, ...,u_K> to denote the basic structure.

The key management algorithm we use for the basic structure is the complete key graph algorithm. In this algorithm, for every nonempty subset of users, the group controller provides a unique shared key which is known only to the users in the subset. The group controller gives these keys to the users at the time of joining the group. Of the keys that a user, say u_i, receives: 1) one key is associated with the set { u_1, u_2, ...,u_K}, and hence, is known to all the users and 2) one key is associated with the set {u_i} The former key, say k_R, is the group key, whereas the latter key is the personal key. Thus, the number of keys stored by the group controller is 2^K-1 and the number of keys held by each user is 2^{K-1}. Now, we consider the process of rekeying in this scheme when one or more users are revoked from the group. The proof of the following theorem describes the simple rekeying process for user revocation:

3.2The Hierarchical Key Management Algorithm

In our hierarchical algorithm, we compose smaller basic structures in a hierarchical fashion. To illustrate the hierarchical structure, consider the sample structure <R, R_1, R_2, . . ., R_d> shown in Fig. 1b, where each further consists of the basic structure <R_i, u_{i1}, u_{i2}, . , u_{id}>. The parameter d is the number of elements in a basic structure and can be considered as the degree of the hierarchy. We note that the degree can be different for different nodes in the hierarchy. However, for the sake of simplicity, in this section, we assume that the nodes in the hierarchical structures have a uniform degree d.

Now, each of the basic structures of the form <R_i, u_{i1}, u_{i2}, . . . , u_{id} > is associated with the shared keys. The structure at next higher level, <R, R_1, R_2, . . ., R_d>, is also associated with shared keys. The personal key associated with Ri, $1 \leq i \leq d$ in structure <R, R_1, R_2, . . ., R_d> is the same as the group key of the structure <R_i, u_{i1}, u_{i2}, . . . u_{id}>. Furthermore, the structure <R,R_1,R_2 . . .R_d> is associated with shared keys. Now, each user in the basic structure <R, u_{i1}, u_{i2} . . . u_{id} > is provided with any shared key that is provided to Ri in the structure <R, R_1,R_2, . . .R_d>. To illustrate our hierarchical algorithm, we consider four examples for d = N, 2, 3, 4. In the hierarchical structure, we denote the key associated with a subset <a, b . . .z> by $K_{ab\ldots Z}$.

III. MEMBERSHIP ADDITION COST

When users get revoked from a hierarchical structure, it does not change the number of keys that the existing users would have although some of the keys that they maintain may no longer be needed. For example, suppose we begin with a basic structure of degree 4 where each user has eight keys. If u_4 is revoked, then the basic structure still has a degree 4 but it now has one empty slot. The remaining users continue to have eight keys although some keys are currently useless. When a new user is added to this structure at a later point, these keys would be updated and the revised keys would be given to the new user.

Adding users to a basic structure with empty slots. We first describe the algorithm where enough empty slots are available in the hierarchical structure. The procedure for adding a user to the group is as follows: the group controller changes the group key and distributes it to the current users of the group and to the joining user. The group controller also distributes the necessary keys to the joining user. If multiple users are to be added, the group controller generates the new group key and distributes it to each of the new users in a separate unicast message which also contains other keys that are needed by that user. First, the group controller selects a basic structure with empty slots in the hierarchy and adds the new user to this basic structure. Next, the group controller generates a new group key and distributes it to the current users using by encrypting it with the old group key. In this message, it also notifies the users about the location of the new user. Subsequently, the current users use the following rule to generate the keys that are given to the new user is the new group key, f is a one-way hash function, and k_i is the key that is known to any current user that is part of the same hierarchy as the joining user. The group controller also performs the same hash computation and identifies the shared keys that the joining user would get based on its position in the hierarchical structure. (Note that the joining users only get the updated shared keys, and hence, cannot compute old shared keys.) Subsequently, the group controller sends the new group key and the updated shared keys to the joining user. It follows that the number of encryptions performed by the group controller is equal to the number of keys that need to be distributed to the joining user, i.e., O(2^{d-1} log(N). Thus, the cost of join is equal to the number of keys that each user maintains.

Our approach of using hash functions to change the group key is used by others [6] and is acceptable even if it increases collusion possibility as described. Though, it is possible to address collusion by changing the intermediate keys explicitly, this increases the join cost significantly. However, one important advantage is that the cost of join is non critical using our algorithms, i.e., the join handling do not have to be performed immediately, whereas revocation has to be performed immediately. Join can be handled in background as the new user does have the group key right away.Increasing the height of the hierarchy. Adding a level in the hierarchy is straightforward. Let T be the current tree with root node R. To add a level in the hierarchy, we can create a new root R_1, let R be its child and create additional children for R_1. Note that in this case, users can continue to keep the keys that they have. They need to receive additional keys for this new hierarchy. However, similar to join process for a basic structure with empty slots, the cost of increasing the height of hierarchy is small.We do not propose new schemes for reducing the height of hierarchy when users leave. Most group key management algorithms utilize periodic rekeying to deal with lost keys, collusion, etc. At the time of periodic rekeying, the height can be reduced using techniques similar to [4].

IV. ADAPTING TO HETEROGENEITY OF USERS

Our algorithms also enable the group controller to deal with heterogeneous set of users who have different capabilities. We illustrate this by a simple example. Consider the case where the basic structure at the root level has a degree 2, the users rooted at the left child of the root can only maintain a small number of keys, and the users rooted at the right child of the root can maintain a large number of keys. Now, we can use a smaller degree for the tree rooted at the left child and a larger degree for the tree rooted at the right child. With such a design, the users in the left tree will receive only a small number of keys, whereas the users in the right tree will receive a large number of keys. It follows that for the right tree, the group controller can take advantage of reduced rekeying cost provided by the use of a tree with larger degree, while still allowing users with lower capabilities to participate in the group communication.

Based on the above discussion, we can use a higher degree for the basic structure at the root to accommodate users with multiple storage requirements. In such a key tree structure, the users rooted at each child node have the same requirements and capabilities. Thus, by partitioning the group at the basic structure, the group controller can deal with heterogeneous users in a fine-grained manner. Specifically, we examined two cases of variations in the degree of the key tree. In each case, the group controller maintains a key tree of different degrees for two or more child nodes of the root node. We evaluate our algorithms using simulations on groups of size 256, 512, and 1,024 users.

V. REDUCING STORAGE REQUIREMENTS FURTHER

we provide additional approaches for reducing the storage requirements of users. The first approach is based on [8] and is aimed at providing adaptation where long-term users are provided preferential treatment in that they store less keys and need to perform less computation when group membership changes. The second approach, reduces the storage by permitting users to generate shared keys using their personal keys. This scheme is suited in situations where group revocations occur at some periodic times. For brevity, we only present the scheme for the basic structure. It can be extended in a straightforward manner for hierarchical structure.

‘

6.1 Adapting for Long-Term and Short-Term Users

Key assignment technique to adapt to the storage requirements of long-term and short term users. Long-term users are those users who have been in the group a relatively longer period than the other group users. Short-term users are users who have been in the group for a relatively short time. In our key assignment, we assign keys in such a way that the longer a user stays in the group, the number of keys stored by that user is relatively smaller than the keys stored by short-term users. We can view this result as rewarding long-term users for their long standing membership.

One-way chains are of the form h(s), $h^2(s)$, . . . $h^m(s)$, where h is a one-way hash function and s is a random seed. Hence, using an intermediate value $h^k s$ in this chain, the higher values $h^{k+}1(s)$, $h^{k+2}(s)$ . . . , in the chain can be generated by using h. Furthermore, due to the one-way nature of the hash function, by knowing an intermediate value $h^k s$, it is not possible for a user to deduce the previous values $h^{k-1}{}_{(S)}$, $h^{k-2}(s)$ . . . , in the hash chain. The above hash chain can be trivially extended to the case where different one-way functions are used in each step. In this case, the hash chain would be of the form $h_1(s)$ $h_2(h_1(s))$, $h_3(h_2(h_1(s)))$ … . . . ,. Note that even in this case, if a node has a value in this hash chain, then it can find all subsequent values. However, it cannot find previous values in the hash chain.

Now, using these concepts, we describe our technique for arranging the keys among the users. For a group of d users u_1, u_2; . . . , u_d, we use d - 1 (or more) hash functions h_1,h_2 . . ,. Now, consider a user subset {u_a, u_b, u_c}, where a < b < c. For such a subset, we consider the chain <u_a, u_b, u_c>. For such a chain, we assign secrets as follows: u_a is assigned a seed secret s_a u_b is assigned secret $h_{b-a}(s_a)$, and u_c is assigned $h_{c-b}(h_{b-a}(S_a))$. Thus, the secret provided to u_c (respectively, u_b) can be used for communicating with the set {a, b, c} (respectively, {a, b}. For example, in set {u_2, u_3, u_5}, u_2 will get s_2, u_3 will get h_1S_2, and u_5 will get $h_2(h_1s_2)$. Thus, by having only a small set of secrets (some), users can generate secrets needed for different subsets.

However, a single one-way hash chain is not sufficient to assign keys to every possible subset of the users. For example, in the above scenario, there is no unique secret for the set f(u_a, u_c). Since the basic key structure requires that a key be maintained for each subset of users, there is a need for additional one-way hash chains. Hence, there is need for multiple one-way hash chains to assign keys to all possible subsets of users. Next, we present our key assignment technique that assigns keys to every possible user subset.

Our key assignment is inductive in nature. It also has the property that in any chain, the users are labeled in increasing order. Moreover, the last user added to the set is present in every chain. For n = 1, i.e., where there is only one user, say u_1, there is only one chain (u_1). For inductive case, assume the key assignment for n users and we need to obtain the key assignment for n +1 users. Let u_n be the user with the highest label in the existing system and u_{n+1} be the new user. Now, based on our assumptions, u_n is the last user in all the one-way hash chains formed for the set of n users. The list of hash chains for n +1 users is obtained as follows:

- For each hash chain, include a hash chain where u_{n+1} is added to the end of that hash chain.
- For each hash chain, include a hash chain where u_n is replaced by u_{n+1}.

Adaptive storage. Consider the key distribution for a group of n users, say $u_1, u_2, \ldots, u_n$. Observe that u_1 only needs to store one key, namely, its seed s_1. User u_2 needs to store two keys s_2 and $h(s_1)$. User u_3 needs to store four keys, and so on. Thus, storage of users added earlier is less compared to that of recently added users.

Revocation of users. When a user, say u_m um, is revoked, the keys known to u_m cannot be used. However, the remaining users can continue to communicate using their secrets that were not known to u_m. Since the original setup ensures that given any subset of users, there exists a key that is known to all of them, this property continues to be true of remaining users as well. We leave it to the reader to verify that the adaptively properties continue to be satisfied, i.e., the user with the smallest label will have one key, the next will have two, and so on. Furthermore, observe that the set of keys that users have is exactly those that they would have if we begin with a group consisting of the smallest logical label and continue to add users based on their increasing label.

Note, however, that with this approach, even if a user is revoked, its logical label cannot be used when a new user is added subsequently. Instead, when a new user is added, its logical label should be larger than all existing users. For example, if the current group is $\{u_1, u_2, u_3\}$, user u_2 leaves, and a new user is added, it should be given a logical label of u_4. number of keys that users need to maintain. However, this requirement necessitates the need for For example, for the set $\{u_1, u_3, u_4\}$, hash function h_3 is needed for the one-way chain $<u_1, u_4>$. For this reason, we have included additional hash functions. If periodic rekeying is used to rekey all the keys in the system, then the user numbering can be restored after periodic rekeying. Furthermore, periodic rekeying would also assist in hierarchical setting. In particular, with hierarchical structure, the group controller could change the basic structures to which users belong. By changing the basic structures in this manner, it would be possible to provide additional trade-off between keys maintained by users and the length of time they are part of the group.

6.2 Another Approach Using a Family of One-Way Functions

We give an additional scheme where group revocations occur at periodic times, for example, once a day. Hence, after one rekeying, there is substantial time. Hence, in such an approach, we can have the rekeying cost split into a critical cost and no critical cost. Our technique has the following attractive features:

- The group controller has to store only N keys, one for each user. The remaining keys are generated using these values. The cost of storage for the group controller using this scheme is lower than that of LKH, where 2N - 1 keys need to be stored and that of the complete key tree algorithm where 2^d.N keys have to be stored.

- The cost of storage at the users is reduced by a factor of 2 when compared the storage required for the complete key tree algorithm. The revocation cost remains the same as the key distribution is essentially our original complete hierarchical structure.

- The cost of updating the shared keys during a membership change is O(logN) messages for the group controller.

We introduce additional notation used in this technique. We use g_i to denote the ith member of a family of one-way functions g. The property of a one-way function is such that, given x, it is easy to compute g(x) but not the vice versa. When a one-way function g is applied to a key k, we say that the resulting value is a blinded value of that key. Blinded keys that need to be given to a user i are blinded using the one-way function g_i.

VI. SECURITY MODELS FOR OVERLAY NETWORKS

The system is designed to manage storage and key revocation process. Multicast group and key management operations are integrated in the system. The system is enhanced to manage group keys under overlay network environment .The key management messages are controlled in the system

The system is divided as five modules. They are overlay network construction, storage management, key management, rekeying management and transmission controller. The system is designed to handle group communication with security under overlay networks. The storage overhead is managed by the system using lifetime factors. The key issue and key revocation operations are managed by the system

7.1. Overlay Network Construction

The wireless nodes are grouped to form overlay networks. The overlay networks are used to extend the coverage of the network .The node to node communication is used in the overlay networks. Neighborhood verification is used for overlay updates

7.2. Storage Management

The key values are maintained under the node storage area .The storage is updated with other nodes key value during key revocation process .The lifetime is considered in the key update process .Each node maintains key values for different set of nodes

7.3. Key Management

The group key values are maintained in the network. The multicast group nodes are assigned with the same key values. Each node maintains two level keys .Node key and group key are used in the system

7.4. Rekeying Management

The rekeying process is done at group updates. The key revocation is initiated at the time of node entry and node removal. The key revocation process updates key values of the entire group. The system reduces the rekeying intervals

7.5. Transmission Controller

The data transmission activities are managed under the transmission controller. Key request and key revocation operations are initiated by message communication .The key request messages are limited by the system .The key value is used in the data security process

VII. CONCLUSION

The secure multicast transmission supports key management under multicast groups. The storage and key revocation operations are managed by the system. A family of algorithms is used to provide a trade-off between the number of keys maintained by the users and the time required for rekeying due to the revocation of multiple users. The algorithms reduce the cost of rekeying. The schemes are based on the use of one-way hash chains that allow one to reduce the number of keys further without increasing the rekeying cost.

The algorithm enables the group controller to deal with heterogeneous set of users that have different capabilities. With this capability, users with high capability can benefit from it. The system is enhanced to manage keys under overlay networks. The traffic and energy control mechanisms are integrated with the system. The system supports overlay network multicast process. Storage usage is controlled by the system. Traffic levels are managed by the system. The system reduces the energy levels.

AUTHOR'S PROFILE

P.G.Kathiravan received the B.Tech in Information Technology from The Kavery Engineering College in Anna University, Chennai in 2010. He is currently doing M.Tech Information Technology from K.S. Rangasamy College of Technology (Autonomous), Anna University, Coimbatore. His area of interest is Network Security, Cloud Computing, and Operating System

C.Rajan received his B.E Degree in Computer Science and engineering from SSN College of engineering at University of Madras. Then he obtained his Master's degree in Computer Science. He is pursuing Ph.D at Anna University of Technology, Coimbatore. He is currently working as an Assistant Professor in the Department of Information Technology, KSR College of Technology. He has 8 years of teaching experience. He has presented 06 papers in various national and international conferences. His research interests Multicasting Networks, Key Management and Network Security.

Dr.N.Shanthi received the Ph.D. degrees in computer sciences and Engineering from the Periyar University, Salem, India. She is the professor and head of Department of Information Technology, KSR College of Technology. She has 18 years of teaching experience. She has more than 13 publications at national and international level. Her areas of interest include Network security and Image Processing.

REFERENCES

[1] Y. Kim, A. Perrig and G. Tsudik, *"Tree- Based Group Key Agreement,"* ACM Trans. Information and System Security, vol. 7, no.1, pp.60-96, 2004.

[2] M. Manulis, *"Security-Focused Survey on Group Key Exchange Protocols,"* Report 2006/395, Cryptology ePrint Archive, http:// eprint.iacr.org/, 2006.

[3] F. Zhu, A. Chan, and G. Noubir, *"Optimal Tree Structure for Key Management of Simultaneous Join/Leave in Secure Multicast,"* Proc. Military Comm. Conf. (MILCOM), 2003.

[4] W.H.D. Ng, M. Howarth, Z. Sun, and H. Cruickshank, *"Dynamic Balanced Key Tree Management for Secure Multicast Communications,"* IEEE Trans. Computers, vol. 56, no. 5, pp. 577-589, May 2007.

[5] S. Zhu, S. Setia, S. Xu, and S. Jajodia, *"Gkmpan: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks,"* Proc. IEEE Mobiquitos '04, pp. 42-51, 2004.

[6] Y. Sun, W. Trappe, and K.J.R. Liu, *"A Scalable Multicast Key Management Scheme for Heterogeneous Wireless Networks,"* IEEE/ACM Trans. Networking, vol. 12, no. 4, pp. 653-666, Aug. 2004.

[7] M.H. Heydari, L. Morales, and I.H. Sudborough, *"Efficient Algorithms for Batch Re-Keying Operations in Secure Multicast,"* Proc. 39th Ann. Hawaii Int'l Conf. System Sciences, vol. 9, 2006.

[8] J.H. Cheon, N. Jho, M. Kim, and E. Yoo, *"Skipping, Cascade, and Combined Chain Schemes for Broadcast Encryption,"* IEEE Trans. Information Theory, vol. 54, no. 11, pp. 5155-5171, Nov. 2008.

[9] Bezawada Bruhadeshwar and Sandeep S. Kulkarni, *"Balancing Revocation And Storage Trade-Offs In Secure Group Communication"* IEEE Transactions on Dependable and Secure Computing, Vol. 8, no. 1, Jan-Feb. 2011.