IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-8, Issue-1, Jan 2020 E-ISSN: 2347-2693

Ensuring Secured Multicast Group Communications for Wireless Sensor Networks using Efficient Key Distribution Schemes

S. Sasikala Devi

PPG College of Arts and Science, Saravanampatti, Coimbatore.

Author:sasishrinithi.s@gmail.com

DOI: https://doi.org/10.26438/ijcse/v8i1.120126 | Available online at: www.ijcseonline.org

Accepted: 17/Jan/2020, Published: 31/Jan/2020

***Abstract* –** Multicast is the only prominent method for transmitting data from a single source to several known destinations. More than ever, in wireless sensor networks, with the help of unguided medium, a single transmission able to be received by all nodes within a transmission range.

A wireless sensor network (WSN) are spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. For that reason, the multicast in wireless networks is anticipated to lay concrete on the way for efficient group communications, by which many group-based applications, such as charged video on demand or video conferencing, can be commercialized. In WSNs security, the key management problem is one of the most important and the most fundamental aspects. To attain security in wireless sensor networks, it is significant to be able to encrypt and authentication messages among sensor nodes. Before doing so, keys for performing encryption and authentication must be agreed upon by the communication nodes among the WSN. Nevertheless, due to the resource constrains on the sensor nodes, many key agreement mechanisms used in general networks, such as Diffie-Hellman and other public-key based schemes , are not feasible in sensor networks.

Keywords: WSN, Multicast Rekeying, key tree, Accuracy, Computation and Communication time.

I. INTRODUCTION

An effective key management scheme is the basis of the other security mechanism such as secure route, secure localization, confidentiality, authenticity, availability, and integrity. Recently, the key management problem has been extensively studied in the context of WSNs. The low memory and energy physical constraints of sensor nodes limit key management scheme in the real world.

Sending data over the unguided medium is really a risky job since hackers over the network are capable enough to take away the data. This motivates this research work. Secure group communication relies on secure and robust distribution of group keys. A single symmetric key known only to the group members can effectively protect a multicast group. However, only legitimate users should have access to the group communication in order to achieve privacy. Thus the group key (session key) must be updated each time when new users join or old users leave the group and securely redistributed to the existing members of the group. This is referred to as group rekeying. The newly joint users should not be able to derive the previous group keys, even if they are able to derive future group keys with subsequently distributed keying information.

Similarly, the revoked users should not be able to derive the future session keys, even if they are able to compute the previous session keys with previously distributed keying information. If a group is rekeyed on each membership change, the frequency of rekeying becomes the primary bottleneck as the size of the group grows and/or the rate of membership change increases. Thus, scalable group rekeying is an important and challenging problem to be addressed in order to support secure multicast communication for dynamic groups, where typical systems are large: tens of millions of users. Distribution of key efficiently over an unreliable channel is an interesting research topic.

The objectives of the research are as follows

- To propose key distribution schemes for scalable wireless sensor networks.
- To model the propose schemes for managing key distribution in a distributed manner.
- To conduct performance analysis of all the proposed key distribution algorithms in terms of accuracy, computation time, recovery time, key distribution time and communication cost.

II. LITERATURE REVIEW

Security is essential for data transmission through an insecure network. There are several schemes to address the unicast security issues but they cannot be directly extended to a multicast environment. In general, multicasting is far more vulnerable [Peter Kruus and Joseph Macker.,1998, Paul Judge and Mostafa Ammar.,2003, Moyer et al.,1999] than unicast because the transmission takes place over multiple network channels. In multicast group communication, all the authorized members share a session key, which will be changed dynamically to ensure forward and backward secrecy referred as "group rekeying". The forward secrecy ensures that the members who left the group cannot get access to future group data, and the backward secrecy ensures that currently joined members cannot access past group data. For a multicast group with a large number of members, key-tree based schemes were introduced to decompose a large group into multiple subgroups with smaller sizes [Waldvogel et al.,1999, Mittra.,1997, Wallner et al.,1998, Wong.,1998]. For stateful MKD protocols, assignment of personal keys to a joining user is on-the-fly, and usually based on some kind of dynamically changing group access structures. Typical group access structures used by stateful MKD protocols are logic key hierarchies [Wong et al.,2000, Wallner et al.,1998, Caronni et al.,1998, Canetti et al.,1999, Perrig et al.,2001, Waldvogel et al.,1999], bottom-up one-way function tree [Sherman and McGrew.,2003, Liu and Yang.,2011], flat table [Waldvogel et al.,1999, Chang et al.,1999, Zhou and Huang.,2010], dual hash chain [Fan et al.,2002], [Liu and Wang.,2011], and top-down one-way function tree, [Liu and Wang.,2011]. When a user joins the group, GC needs to first create one or several new nodes/shares on these group access structures for it, and then update relevant keys before assigning them to the joining member to ensure group backward secrecy. When a user leaves the group, GC needs to delete its associated node from these group access structures, and then update those keys held by the evictee (thus all information about the group access structure held by the evictee is invalidated) to ensure group forward secrecy. In a word, these group access structures keep expanding, contracting, and changing as members join or leave. Their current size exactly corresponds to the current number of group members. That is why we say they are dynamic. Therefore, most stateful MKD protocols based on dynamic group access structures are more suitable for immediate rekeying for large and dynamic groups than those stateless protocols based on static group access structures.

III. PROPOSED METHODOLOGY

A. Optimized PFMH Tree Based Multicast Rekeying for Wireless Sensor Networks

1. Improved Maximum Distance Separable Codes

Improved Maximum Distance Separable (MDS) codes are a class of error control codes that meet the Singleton bound. Letting GF(q) be a finite field with q elements an (n,k) (block error) control code is then a mapping from $GF(q)^k$ to $GF(q)^n$:E(m)=c, where $m=m_1,m_2,...,m_k$is the original message block, $c=c_1,c_2,...,c_n$ is its code word block and E(.) is an encoding function, with $k \leq n$ If a decoding function D(.) exists such that $D(c_{i_i}, c_{i_2}, \dots, c_{i_k}, i_1, i_2, \dots, i_k) = m$ for $1 \leq i_j \leq n$ and $1 \leq j \leq k$, then this code is called an (n,k) Improved MDS code. For an (n, k) Improved MDS code, the k original message symbols can be recovered from any k symbols of its code word block. The process of recovering the k message symbols is called erasure decoding. All the symbols are defined over GF(q), and usually, $q = 2^m$. The well-known Reed-Solomon (RS) codes are a class of widely used MDS codes. It is to be noted that the proposed Improved MDS code can be used to construct secret-sharing and threshold schemes using PFMH key structure.

2. PFMH Key Tree Structure and Basic Procedures

In tree-based contributory group key agreement schemes, keys are organized in a logical tree structure, referred to as the key tree. In a key tree, the root node represents the group key, leaf nodes represent the members' private keys, and each intermediate node corresponds to a subgroup key shared by all the members (leaf nodes) under this node. The key of each non leaf node is generated by making use of improved MDS between the two subgroups represented by its two children where each child represents the subgroup including all the members (leaf nodes) under this node. Since the two-group is formed using improved MDS, the key tree is logical. The insertion of node will be logically carried out in the right side of the key structure.

In this scheme, each member will maintain and update the global key tree locally. Each group member knows all the subgroup keys on its key path and knows the ID and the exact location of any other current group member in the key tree. In PACK, when a new user joins the group, it will always be attached to the root of the join tree to achieve O(1) rekeying cost in terms of computation per user, time, and communication. When a user leaves the current group, according to the leaving member's location in the key tree, as well as whether this member has a phantom location in the key tree, different procedures will be applied, and the basic idea is to update the group key in O(log n) rounds and simultaneously reduce the communication and computation costs.

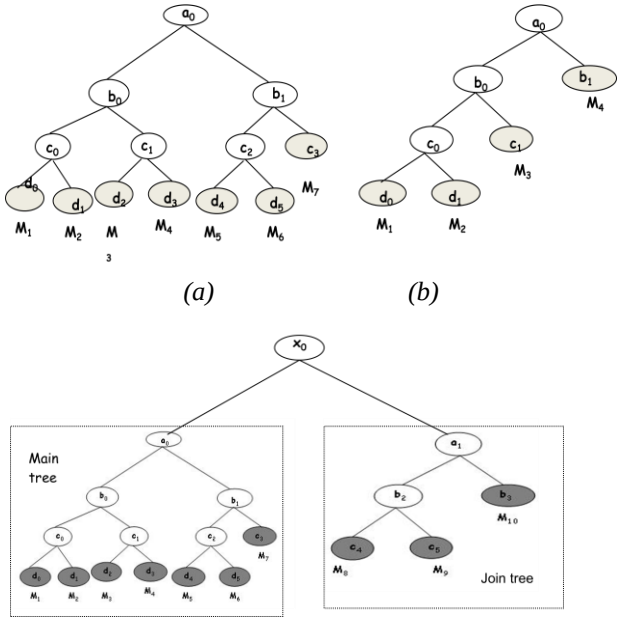
3. PACK: A PFMH Tree-Based Contributory Group Key Agreement

PFMH tree is an efficient logical key tree structure for contributory group key agreement schemes. PFMH tree is a combination of two special key tree structures: *partially full* (PF) key tree and *maximum height* (MH) key tree. In this

paper, the total number of leaf nodes indicates the size of a key tree. The function *log()* and $log_2()$ will be used exchangeable, and if it is a "full (key) tree," it mean a fully balanced binary (key) tree with size 2k, where k is a nonnegative integer. Fig.1, shows the example of PH, MH and PFMH key tree.

PF key tree: Let T be a binary key tree of size n, and $n' = 2^{[\log n]}$. The left subtree of T is a full key tree with size n', and the right subtree of T is a PF key tree with size (n-n').
MH key tree: The right subtree of T is a leaf node, and the left sub tree of T is an MH key tree with size n - 1.
PFMH key tree: The left subtree of T is a PF tree, and the right subtree of T is an MH tree.

The height of a PF key tree with size *n* is *[log n]*, the height of an MH tree with size *n* is n-1. In PFMH tree T, *main tree* refer to the PF subtree, T_{main} and *join tree* refer to the MH subtree of T, and Tjoin. There are two basic procedures to manage and update PFMH tree they are *unite* and *split.*



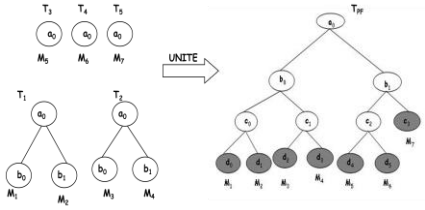
(a) *(b)*

(c)

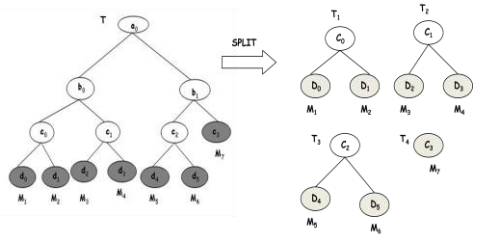
Fig 1 .Examples of a)PF b) MH c) PFMH key tree.

1) Unite Procedure: Let $T = \{T_1 \ldots T_L\}$ be a set of full key trees. Each key tree $Ti \varepsilon T$ represents a subgroup, and each leaf node of *Ti* is a member of this subgroup. If a group member belongs to *Ti* and $Ti \varepsilon T$, then this group member belongs to *T*. The procedure *unite(T)* is to combine those key trees in *T* into a single PF key tree.

2) Split Procedure: Given a key tree *T*, the procedure *split (T)* is to partition *T* into a set of full key trees with the minimum set size. This Procedure presents a way to locally and virtually *split* a key tree, where "locally" means that no intercommunication is needed among group members and each member only needs to update the key tree structure maintained by itself locally, whereas "virtually" means that no two-group is needed to perform "split." Fig.2 shows example of key tree update after applying a) unite and b) split procedure.



(a)

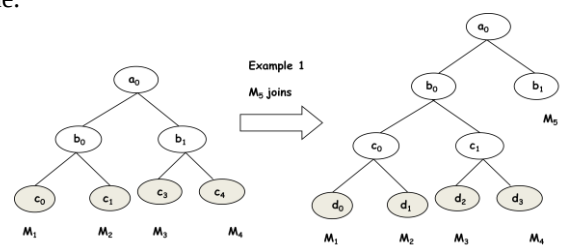


(b)

Fig.2 Examples of a key tree update after applying a) unite and b) split procedures.

4. Single-User Join Procedure

When a new user *M* wants to join the group G, the PACK initiates the single-user join protocol by broadcasting a request message that contains its member ID, a join request, its own blinded key, some necessary authentication information, and its signature for this request message. After receiving this user join request message, the current group members will check and a new group key will be generated in order to incorporate a secret share from *M. F*ig3 depicts the same and to guarantee the group keys' backward secrecy. The rekeying upon single-user join needs to perform two rounds of improved MDS code.



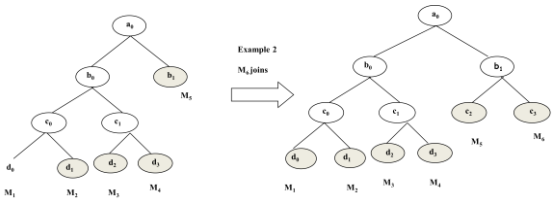


Fig.3 Examples of a key tree update upon a single-user join event.

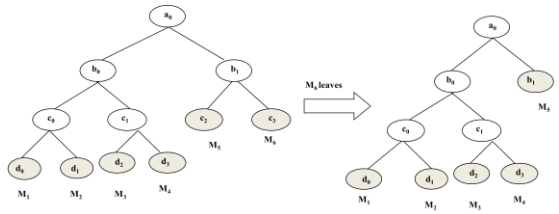


Fig 4. Update upon single-user join events

Fig.3 shows two examples of a key tree update upon single-user join events. In the first example tree consists of four members. After the new member M5 joins the group, a new node is created to act as the new root, and the node (b1) becomes the new join tree that represents M5. In the second example, when M6 joins the group, at the first round, the MDS is first performed between M5 and M6 to generate a new join tree, at the second round, the MDS is performed between the new join tree and the main tree to generate a new group key.

5. Single-User Leave Procedure

When a current group member Y wants to leave the group, it broadcasts a leave request message to initiate the single user leave protocol, which contains its ID, a leave request, and a signature for this message. In order to reduce the rekeying cost upon a single-user leave event, PACK creates a phantom node that allows an existing member to simultaneously occupy more than one leaf node in the key tree. Fig 4 shows one example of a key tree update upon single-user leave event. In this example, user M6 leaves the group where node (b0) is the root of the main tree and node (b1) is the root of the join tree. Since the size of the join tree is 2, the node representing M6 will be directly removed from the key tree, M5 changes its secret share, and a new group key will be generated by applying the MDS between M5 and the subgroup in the main tree.

6. Group Merge and Group Partition Protocols

PACK also has group merge and group partition protocols to handle simultaneously the join and leave of multiple users. Although multiple user events can be implemented by applying a sequence of single-user join or leave protocols, such sequential implementations are usually not cost-efficient. The group merge protocol, combines two or more groups into a single group, and returns a PF key tree. Group partition protocol, removes multiple group members simultaneously from the current group and construct a new PF key tree for the rest of the group members. In the group merge protocols, after removing all phantom nodes from those key trees corresponding to different subgroups, each key tree is split into several full key trees. The final result is obtained by uniting these full key trees into a PF tree using unite procedure. Similar to the group partition protocol, after removing all phantom nodes and leaving nodes, the original key tree is split into several full key trees, and the unite procedure is then applied on these full key trees to create a PF key tree. Since the height of the returned tree is log n, where n is the group size after merging/partitioning, the time cost of group merge/partition is bounded by *O(log n)*.

B. Conventional Approach Probability based PFMH Tree Based Multicast Rekeying for Wireless Sensor Networks

As mentioned in [Yu et al. 2007], it is said that Cauchy-matrix-based RS codes are considered as more efficient than Vandermonde-matrix based RS codes since it requires less complexity than that of Vandermonde-matrix based RS codes. Quite contrary, it is observed that Vandermonde representations are, in fact, more efficient in terms of decoding operations for the rekeying process. The main reason is that the inverse Vandermonde matrix is much simpler than the inverse Cauchy matrix. Based on the above observation, Vandermonde matrix is used to construct RS codes, as conventional wisdom suggests.

The probability based PFMH (PPFMH) tree based technique is proposed for the environment with dynamic users. In the previous PFMH method, the key-tree optimization algorithms preserves as balanced and complete a tree structure as probable after group membership changes (joining or leaving) cannot accomplish optimal performance in reality. But the proposed method adopts the circumstances like random number of members, non-equal leaving probabilities, and non-balanced and non-complete tree structure. So, it accomplishes optimal performance in reality. So, during the group membership changes, the average communication over head of a key-tree is less. This method comprises the creation of basic tree and for optimization of key tree after membership changes. This method facilitates to create an optimal key-tree that replicates the uniqueness of users' leaving probabilities, and facilitates repeated preservation of communication with less overhead in group rekeying. The leaving probability is defined as an average number of users leaving in the group in a particular rekey interval. In this method, the probability is calculated as the ratio of number of users leaving in the group and number of rekey intervals in which the user subscribes. Already in the network the group of users is deployed. We assume the joining and leaving users based on the probability. So, based on the probability the keys are generated. Suppose, if the

new users are join to the network the already generated keys are given to the users. Through simulations, we show that Probability based PFMH tree (PPFMH) framework outperforms the previous one which is known to be the best balanced and complete structure. By using this Probability based PFMH tree (PPFMH) for key distribution scheme there is less computation and communication complexity.

The leaving probability is determined as,

$$p(u) = \frac{Number\ of\ u\ leaving's\ the\ group}{Number\ of\ rekey\ intervals\ in\ which\ u\ subscribes}$$

For the new user, however, because the numerator and the denominator are 0, we cannot obtain its leaving probability. In this situation, to set the probability as the average leaving probability of the whole users.

In this case, we set the probability as the average leaving probability of the entire users. The probability value of the particular user is high, the user leaves and joins more recurrently. This leaving probability decides the structure of the optimal key tree.

The number of users is initialized in the network. The initial trees is constructed and determine the leaving probability of the users. The users are sorted in growing order of leaving probability $p(u)$. The remaining probability of the users is also determined for a particular rekey interval. The algorithm consists of two functions called split key and create child. The traffic encoding keys are used to share between the server and all users in the group. The key encoding keys are used to share between the key server and the corresponding user. This algorithm has two steps: One is remove and place users and re-structure the key tree. This method eliminates the leaving users and places recently joined users by using the growing order of users' leaving probabilities. The parent node for the leaving and joining users are marked. Suppose, if the marked parent node has only leaving users, the node is marked as -1, 1 for the joining users and 0 for both leaving and joining users. If the node is marked as 1, the number of child key-nodes of the marked node increases and if the node is marked as -1 the number of child key-nodes of the marked node decreases. At last, the leaving probability is updated.

1. Leaving probability

The leaving probability is calculated as the average leaving probability of the entire users. Suppose, in the first rekeying interval RI_1, totally five users are left from the network. In the second rekeying interval RI_2, totally 7 users are left from the network. So, the leaving probability is calculated at particular rekeying interval, by taking the average leaving probability of the entire users.

7+5/2=6

So, at a particular rekeying interval there are six users leaving from the network. From this we calculate the leaving probability. The remaining probability of the users is also determined for a particular rekey interval. The joining and leaving users are determined based on the probability. So, based on the probability the keys are generated.

2. Joining Probability

The joining probability is also computed as the average joining probability of the entire users.

Suppose, in the first rekeying intervalRI_1, totally five users are joined in the network. In the second rekeying intervalRI_2, totally 7 users are joined in the network. So, the joining probability is calculated at particular rekeying interval, by taking the average joining probability of the entire users.

7+5/2=6

So, at a particular rekeying interval there are six users are joined in the network. From this we calculate the joining probability. The remaining probability of the users is also determined for a particular rekey interval. The joining and leaving users are determined based on the probability. So, based on the probability the keys are generated.

C. An Efficient Cluster PFMH Tree Based Multicast Rekeying for Wireless Sensor Networks

An important problem for secure group communication is key distribution. Most of the centralized group key management schemes employ high rekeying cost. Here, introduce a novel approach for computation efficient rekeying for multicast key distribution. This approach reduces the rekeying cost by performing a hybrid key distribution scheme by combining both PFMH and PPFMH key management schemes. The member in the group uses the improved MDS Codes, a class of error control codes, to distribute the multicast key dynamically.

Since, the number of leaves determines the total number of nodes in a tree of given degree, if set the number of leaves as a variable, then control the total number of keys. One approach is to cluster the members and assign multiple members to a leaf, then by controlling the number of members assigned to a leaf node, it can vary the total number of nodes in the tree and thus the number of keys stored in the randomly chosen member node. Then use the hybrid tree model in order to develop the design algorithm for a given amount of update communication.

The main idea of the hybrid tree with PPFMH key is to divide the group into clusters of size M with every cluster assigned to a unique leaf node. Then there are N/M clusters (also leaves), and need to build a tree of depth log a (N/M).
The proposed method of Hybrid tree based key distribution consists of two parts, the logical tree, and the clusters. The

logical key tree is used as inter-cluster key management scheme to limit key update communication, and the minimal storage used as the intra-cluster scheme to reduce the randomly chosen member node's storage requirement. In the hybrid tree presented, a user needs to store a key which required by the logical key tree scheme within the cluster. When a member is deleted, the total number of key update messages, denoted by C , is (a-1) $log\ a$(N/M) within the tree plus (M-1) within the cluster, leading to:
C= (M-1) + (a-1) $log\ a$ (N/M)

The number of keys stored by the randomly chosen member node is computed as the keys on the tree plus seeds for (N/M) clusters, which is:

$$S = \sum_{i=0}^{\log a\left(\frac{N}{M}\right)} a + \frac{N}{M} - \left(1 + \frac{a}{a-1}\right)\frac{N}{M} - \frac{a}{a-1}$$

The last term 1/(a-1) is at most 1 since a≥2.
Since the logical key tree schemes have logarithmic update communication in the hybrid tree model, want to keep the update communication as O(log N) except some scale factor β . This can be expressed as:

$$(M-1) + (a-1)log a\frac{N}{M} \leq \beta log a\ N$$

where the communication scale factor β indicates how much communication can be allotted for key updates. In the hybrid tree scheme, the storage and the update communication are functions of the cluster size M. The selection of M should be such that the update communication scales at least of the order of O (logN) while the key storage of the randomly chosen member node is better than O(N) . Hence the optimization problem is posed as min [(1+ a/a- 1) N/M] w.r.t. M.

With this hybrid tree key distribution the key storage is reduced greater percentage if the total node is in the order of 220. The performance is purely based on the cluster size value M. Then should choose the cluster value based on the applications & security requirements. Within cluster the communication is very much easier than inter communication. And cluster communication provide tight security than the inter communication.

IV. PERFORMANCE METRICS

- Accuracy
- Computation Time
- Recovery Time
- Key Distribution Time
- Communication Cost

V. RESULTS & DISCUSSION

The Accuracy, Computation Time , Recovery Time , Key Distribution Time and Communication Cost of Reed Solomon MDS based Key Distribution, PFMH tree based Key Distribution, PPFMH based Key Distribution, and Cluster based PPFMH based key Distribution is given in the following table.

Table 1 : Comparison of various method

	Accuracy (%)	Computation Time (ms)	Recovery Time (ms)	Key Distribution Time (ms)	Communication Cost (number of multicast)
RS(MDS) [Base Paper]	68%	42.8ms	39.5ms	298.41ms	51
PFMH based Key Distribution (Phase – 1)	72%	38.4ms	27.8ms	241.52ms	48
PPFMH (Phase – 2)	84%	25.3ms	19.2ms	229.16ms	40
Cluster based PPFMH (Phase – 3)	92%	20.9ms	12.4ms	186.43ms	33

VI. CONCLUSION

This research conclude that the proposed cluster based PPFMH tree based key distribution achieves less key distribution time, key generation time and recovery time when compared to rest of the algorithms. Also the accuracy of the proposed protocols are consistently increased and out of the proposed algorithms cluster based PPFMH algorithm outperforms all the other algorithms in terms of accuracy, computation time, recovery time, key distribution time and communication cost.

REFERENCES

[1] Canetti, Garay, Itkis, Micciancio, Naor, Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," Proc. IEEE INFOCOM, pp. **708-716**, **1999.**
[2] Caronni, Waldvogel, Sun, Plattner, "Efficient Security for Large and Dynamic Multicast Groups," Proc. IEEE Seventh Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. **376-383, 1998.**
[3] Chang, Engel, Kandlur, Pendarakis, Saha, "Key Management for Secure Lnternet Multicast Using Boolean Function Minimization Techniques," Proc. IEEE INFOCOM, pp. **689-698, 1999.**
[4] Fan, Judge, Ammar, "HySOR: Group Key Management with Collusion-Scalability Tradeoffs Using a Hybrid Structuring of Receivers," Proc. 11th Int'l Conf. Computer Comm. and Networks, pp. **196-201, 2002.**

[5] Liu and Yang, "Collusion-Resistant Multicast Key Distribution Based on Homomorphic One-Way Function Trees," IEEE Trans. Information Forensics and Security, vol. **6**, no. 3, pp. 980-991, **Sept. 2011.**
[6] Mittra, "Iolus: A Framework for Scalable Secure Multicasting", Proc. of ACM SIGCOMMi'97, **277-288, Sep. 1997.**
[7] Moyer, Rao and Rohatgi, "A Survey of Security Issues in Multicast Communications", IEEE Network Magazine, Vol. **13**, No.6, March **1999**, pp. 12-23.
[8] Paul Judge and Mostafa Ammar, "Security Issues and Solutions in Multicast Content Distribution: A Survey", IEEE Network, **February 2003**, **pp 30-36.**
[9] Perrig, Song, Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution," Proc. IEEE Symp. Security and Privacy, pp. **247-262, 2001.**
[10] Peter Kruus and Joseph Macker, "Techniques and issues in multicast security," MILCOM98,**1998.**
[11] Sherman and McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," IEEE Trans. Software Eng., vol. **29**, no. 5, pp. 444-458, May **2003.**
[12] Waldvogel, Caronni, Dan, Weiler, Plattner,"The VersaKey Framework: Versatile Group Key Management," IEEE J. Selected Areas in Comm., vol. **17**, no. 9, pp. 1614-1631, **Sept.1999.**
[13] Wallner, Harder and Agee, "Key Management for Multicast: Issues and Architectures", Internet Draft (work in progress), draft-wallner-key-arch-01.txt, Sep. **15**, **1998.**
[14] Wallner, Harder, Agee, "Key Management for Multicast: Issues and Architectures," Internet Draft, Internet Eng. Task Force, 1998.
[15] Wong, Gouda and Lam, "Secure Group Communications Using Key Graphs", Proc.ACM SIGCOMM'98, Sep. **1998.**
[16] Wong, Gouda, Lam, "Secure Group Communications Using Key Graphs," IEEE-ACM Trans. Networking, vol. 8, no. 1, pp. 16-30, **Feb. 2000.**
[17] Zhou and Huang, "An Optimal Key Distribution Scheme for Secure Multicast Group Communication," Proc. IEEE INFOCOM, pp. **1-5, 2010.**
[18] MacWilliams , . Sloane, The Theory of Error Correcting Codes. North-Holland Math. Library, **1977.**
[19] Bloemer, Kalfane, Karpinski, Karp, Luby, Zuckerman, "An XOR-Based Erasure-Resilient Coding Scheme," Technical Report TR-95-048, Int'l Computer Science Inst., Aug. **1995.**
[20] Yu, Sun, Liu, "Optimizing the Rekeying Cost for Contributory Group Key Agreement Schemes", IEEE Trans. on Dependable and Secure Computing, vol. **4**, no. 3, pp. 228 – 242, **2007.**

Authors Profile

Dr. S. Sasikala Devi is working as an Assistant Professor & Head, School of Computer Science at PPG College of Arts and Science, Coimbatore, Tamilnadu, India. With a Teaching Experience of about 13 years and Research Experience of 14 years. Her research interest includes Network security, Grid computing, Data mining and Web mining. She has presented 5 papers in an International Conferences and published 9 papers in International journals.

