COLLUSION-FREE MULTICAST GROUP KEY DISTRIBUTION BASED ON TREE STRUCTURES

Syed. Abbas,

M.Tech (CSE),

GVP College of engineering

Email: syed658@gmail.com

P.Aravind(M.Tech),

Associate Professor(CSE),

GVP college of engineering,

Email: aravind53735@gvpce.ac.in

Abstract- **Secure Multicast key distribution (MKD) is very indispensable in all the multicast based applications such as pay TV, Multi-player gaming, video-on-demand etc. These multicast applications require confidentiality and authenticity of transmitted data. Multicast is used to distribute data to a group of receivers efficiently. Group key shared with all authorized senders/receivers is the common solution for controlling the group communication in many multicast applications. There are several tree based MKD protocols such as LKH (logical key hierarchy), OFT (one-way function tree) and HOFT (homomorphic one-way function tree). In these protocols, to update the group key in the event of member eviction or joining, various group rekeying schemes have been proposed. But they are not optimal because of their computational overhead. And OFT has another problem that it is vulnerable to collusion attack.**

In this paper, OFT tree structure with SGRK (Secure Group ReKeying) has been proposed which is secure against collusion attack and has less computational overhead. SGRK provides constant message size and requires O(log n) storage overhead (for a group of 'n' members) at the group server, which makes SGRK suitable for applications containing a large number of group members.

Key words: multicast ,key distribution ,collusion

I.INTRODUCTION

Multicast Group Key Distribution (MGKD) addresses the security problem in open network environments. MGKD restricts the group membership by encrypting the data with a symmetric group key (GK) shared among group members. As the members may join or leave the group dynamically, it is very critical to ensure only legitimate group members have the updated GK at each point in time, which is achieved by GK rekeying. Quite commonly, at the event of membership change, group server generates a new GK and distributes the rekey message to all legitimate members. All members in the group need to be attentive to the rekey messages to update GK. Despite the dynamics of legitimate member set, Group Server wants to preserve backward secrecy[1] and forward secrecy[1], which are defined as follows:

Backward Secrecy: To prevent a new member from decoding messages exchanged before it joins a group, a new group key must be distributed to the group when a new member joins.

Forward Secrecy: To prevent a departing member from continuing access to the group's communication, the key should be updated as soon as a member leaves.

Group key establishment may be broadly categorized as group key exchange/agreement and group key distribution (also called multicast key distribution).In (centralized) multicast key distribution (MKD) protocols, a trusted third party called group server (GS) is responsible for creating a new group key when some change in group membership happens, and securely transferring it to all privileged group members over a broadcast channel. MKD protocols usually aim to solve a more specific problem called immediate group rekeying. For security-sensitive commercial applications (e.g. pay-per-view, video-on- demand,

and highly classified conferences), each message sent to a group is encrypted with a group key, and the group key must be changed for every membership change. To prevent a new member from decoding messages exchanged before it joins a group, a new group key must be distributed to the group when a new member joins. This security requirement is called group backward secrecy. To prevent a departing member from continuing access to the group's communication, the key should be updated as soon as a member leaves. This security requirement is called *group forward secrecy*. To provide both group backward secrecy and group forward secrecy, the group key must be updated (or rekeyed) upon every single change in group membership, and the updated group key must be distributed to all legitimate members. This process is referred to as *immediate group rekeying* in the literature.

II. EXISTING SCHEMES

SKDC (Simple Key Distribution Centre):

It is a linear method for Group Rekeying purpose. A group server shares a secret key with each group member and sequentially uses each member's key to communicate the secret group key to that member. Each time that a member is added to (or evicted from) a group with n members, the group manager must perform n encryptions and transmit n keys. This approach is attractive for relatively small groups. A limitation of SKDC is apparent in the Spread System, which cannot handle groups of more than a few thousand members. By contrast, we are especially interested in handling groups of size 100,000 or more.

Group Diffie-Hellman Method for Group Rekeying:

GDH method offers Distributed functionality, which might be good for some applications, many such proposals have suffered from a linear number of expensive public-key operations. When the network is a tree, however, GDH methods can require only a logarithmic number of operations. Distributed computation is not appropriate for all applications. Furthermore, the basic unit of cost for all GDH methods includes public-key operations, which are slow in software relative to symmetric encryption, one-way function, or pseudorandom function operations.

LKH (Logical Key Hierarchy)

This is the First Tree-based Multicast key distribution protocol. The LKH protocol[2][3] was proposed by Wong And Wallner. Referring to figure 1, each internal node in the key tree represents a key encryption key (KEK), each leaf node of the key tree is associated with a group member, and theroot node represents the group key. KEK is shared by all members associated with its descendant leaf nodes. Every member is assigned to the keys along the path from its leaf to the root. When member leaves the group, all the keys that the member knows should be changed.The group server generates new keys to replace those keys and sends the newly-generated keys encrypted with keys to which the departing member does not have access. If 'n' represents the current number of members in a group and we consider a full and balanced binary tree, leave rekeying using LKH requires at least two O(logn) key encryptions and transmissions by the group server.

When a member joins, the group server creates a leaf node for it, and changes all the keys from this leaf node to the root. In addition to sending the newly-generated keys encrypted with the new member's leaf node secret key, the group

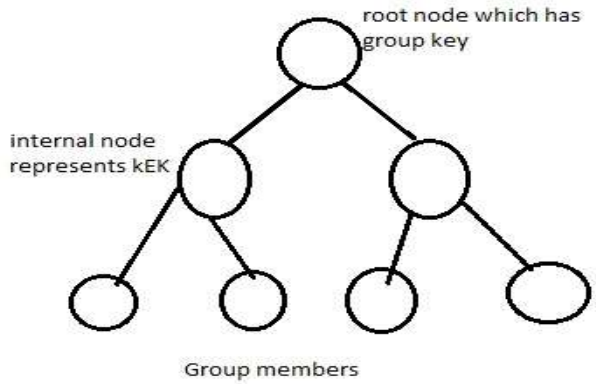

Fig.1: LKH

server sends each new internal node secret key encrypted under the key it is replacing, to the other members. Join rekeying using LKH requires two O(logn) encryptions and transmission of keys by group server.

OFT (One-way function tree)

Group rekeying using One-way Function Tree (OFT) was proposed by Sherman, Balenson and McGrew [4], [5]. The idea of using one-way function (OWF) in a tree structure originated from Merkle. In his report [7], Merkle provided a method to authenticate a large number of public validation parameters for a one-time signature scheme by using a tree structure in conjunction with a one-way and collision-resistant hash function (the famous Merkle authentication tree). A group server maintains a balanced binary key tree for a group. When a member joins/leaves the group, it must update the key tree and distribute a rekeying message to the group to maintain group forward secrecy and group backward secrecy. A one-way function key tree is computed in a bottom-up manner using an OWF (e.g., MD5) and a mixing function (e.g. XOR) as follows. Except the root node, each internal node is associated with two keys: a secret key (also called unblinded key or node secret key in specific context) and a blinded key. The blinded key is computed by applying an OWF (also called blinding function) g to the secret key. Every internal node secret key is computed by applying a mixing function f to the two blinded keys respectively associated with its two child nodes (child blinded keys for short). Like LKH, the group server shares a unique secret key (leaf node secret key) with every group member via a secure channel established during the registration protocol. However, unlike LKH, the group server does not send the members those secrets keys along the path from their leaf node to the root. Instead, it supplies each member the blinded keys of the siblings of the nodes in the path from its associated leaf node to the root of the tree. Each member uses those blinded keys to compute all the secret keys in its path from its parent to the root. Like LKH, secret key associated with the internal node is shared by all members associated with its descendant leaf nodes and the root secret key is the group key.

In LKH, all the keys in the key tree are randomly chosen and thus independent with each other. The hierarchical structure of keys only represents the logical subgroup relationship among the members, that is, key associated with the internal node is shared by all members associated with its descendant leaf nodes. While in OFT, besides the logical subgroup relationship, there is a functional dependency relationship among the secret keys. Functional dependency among keys allows leave rekeying in OFT to save half of communication cost compared to LKH. However, the same relationship also renders it vulnerable to collusion attacks. Different kinds of collusion attacks on OFT are found sequentially by Horng [8] as well as Ku and Chen [9]. Referring to figure 2, suppose that Alice associated with node 8, is evicted at time t_A, and later Candy joins the group at time t_C and it is associated with node 6.We denote the secret key of node i in the time interval between t_A,t_C as $K_i[t_A, t_C]$.

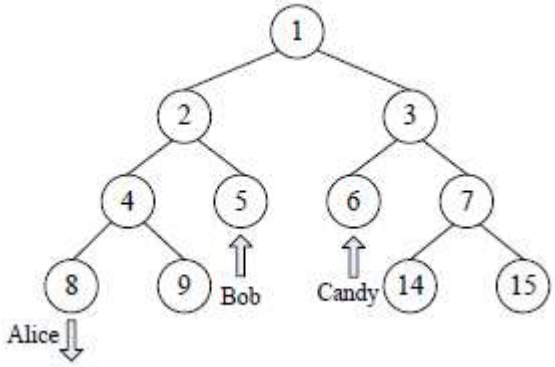

Fig.2 : collusion attack on OFT

The same notation will be used in the rest of this paper. If there are no changes in group membership between time t_A and t_C, according to the OFT scheme, $K_3[t_A, t_C]$ is not affected by the eviction of Alice. The evicted Alice brings out the blinded key $K_3'[t_A, t_C]$ with her. Moreover, the secret key of node two is updated when Alice is evicted, and then remains unchanged even after Candy join.Candy obtains the blinded key $K_2'[t_A, t_C]$ at the time of joining. Collectively knowing $K_2'[t_A, t_C]$ and $K_3'[t_A, t_C]$, Alice and Candy can collude to obtain the group key in the time interval $[t_A, t_C]$ by computing $K_1'[t_A, t_C]$, $K_2'[t_A, t_C]$, $K_3'[t_A, t_C]$. Therefore, the OFT scheme fails to provide group forward secrecy against Alice and group backward secrecy against Candy at the same time. The above attack is due to unchanging keys of the root's children. Horng thus proposed two necessary conditions for a collusion attack to exist: (1) the two colluding nodes must be evicted and join at different subtree of the root; (2) no key update happens between time t_A and t_C. Later, Ku and Chen showed that neither of these two conditions is necessary by proposing two new kinds of collusion attacks.

Improvements on OFT protocol are proposed but at the higher cost of communication overhead. When a new member joins, it will be supplied with the blinded secret keys that were once used to compute the past group key. On the other hand, when a member leaves, it brings out the blinded secret keys that may be used to compute the future group key. It is thus possible for a pair of evicted member and joining member to combine their knowledge together to compute a valid group key between the time of eviction and that of joining. Therefore, it

becomes reasonable to devise a solution to prevent collusion attacks either by preventing a departing member from bringing out any blinded secret keys that contain any information about the future group key or by supplying joining member with blinded secret keys that contain no information about the past group key. Each of the following two improvements on the OFT protocol is just aiming at one aspect to achieve collusion-resistance.

Ku and chen made some assessments and they made some improvements : Ku and Chen improve the OFT protocol by changing all the keys known by a departing member. That is to say, when a member leaves, not only all the secret keys in its path to the root, but also all the blinded secret keys associated with the siblings of those nodes in that path must be changed. The additional updates of secret keys increase the broadcast size by $(\log n)^2$ keys.The group server needs to encrypt and send $(\log_2 n)^2+\log_2 n+1$ keys in total.

Xu observed that collusion between an evicted member and a joining member is not always possible and its success depends on a temporal relationship between them. It is not necessary to always change additional blinded secret keys as above unless a collusion attack is indeed possible. They proposed a stateful approach in which the group server tracks all evicted members and records all the knowledge held by them.

Every time a new member joins, the group server checks against that knowledge to decide whether this joining member could have a successful collusion with any previous evicted member. For that purpose, their protocol has a storage requirement linear to the size of the key tree. The group server will not change additional blinded secret keys as Ku and Chen's protocol until a successful collusion is detected. Therefore, it has a communication overhead lower than Ku and Chen's protocol, but still bigger than the original OFT protocol. In their paper [10], Xu et al. put forward propositions to support the correctness and security of their protocol.

Proposition 1: For the OFT protocol,referring to below figure 3 and table 1,the only secret keys that can be computed by A and C colluding are:

- x_I in time interval $[t_{BMAX},t_{DMIN}]$;
- x_I' in $[t_{BMAX},t_{DMIN}] \cap ([t_A,t_{EMIN}] \cup [t_{EMAX},t_C])$;
- x_I'' in $[t_{BMAX},t_{DMIN}] \cap ([t_A,t_{EMIN}] \cup [t_{EMAX},t_C]) \cap ([t_A,t_{FMIN}] \cup [t_{FMAX},t_C])$; and so on up to the root.

Proposition 2: A pair of colluding members A and C cannot compute any secret key which they are not supposed to know by the OFT protocol, if one of the following conditions holds:

- A is removed after C joins.
- A and C both join.
- A and C are both removed.

NOTATIONS

A,C	Group Members
L,R,I	Left, Right node, Internal node
B,D,E,F	denote the subtrees rooted at nodes L,R,R',R''
t_A	Time that A leaves the Group
t_B	Time that B joins the Group
t_{DMIN}	Time of first group key update happened in D after A leaves the Group
t_{BMAX}	Time of last group key update happened in B before B joins the Group

Table 1: Notations used for generic collusion attack

Proposition 3*:*

For the OFT protocol, an arbitrary collection of removed members and joining members can collude to compute some secret key not already known, if and only if the same secret key can be computed by a pair of members in the collection. Unfortunately, in their proof of this proposition, the authors claim that to compute a secret key not already known, the set of colluding members must already know both child blinded secret keys of it.

This claim is wrong, since the colluding members may not know those child blinded secret keys at first, but can collude to compute them.

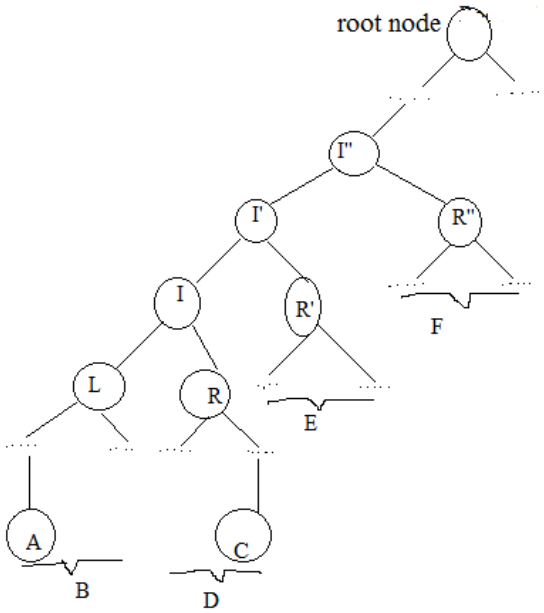

Fig.3: Generic collusion attack

HOFT(Homomorphic OFT):

HOFT [6] is an Enhancement of OFT but it can resist Collusion attacks. However its communication overhead is high when compared to
OFT in the case of join Rekying. Let's get across HOFT with some summarized explanation.

Definition:

A HOFT over an Abelian group (G,*) is a binary key tree that is computed using a self-homomorphic OWF and the multiplicative operation " * " in a bottom-up manner as follows. For an arbitrary secret key x_i in a HOFT 'X' , if x_{2i} and x_{2i+1} are left and right childs of X then we have a Homomorphic property that $x_i = f(x_{2i}) * f(x_{2i+1})$.There are two structure-preserving operations have been proposed with HOFT namely Tree Product and Tree Blinding. A binary operation is said to be Structure-preserving if it takes HOFTs as two inputs and produces HOFT as an output.

Tree product: Given two HOFTs X and Y, both defined over a (G, ∗) and if two trees are isomorphic then tree product of X and Y, denoted by X∗Y, is computed by multiplying their corresponding secret keys.The process is shown in figure 4.

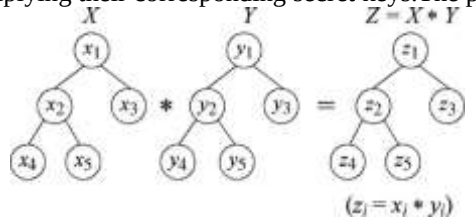

Fig.4: Tree product

Tree blinding: Given an HOFT X, a tree blinding of X maps X to another key tree Y, denoted by Y=f(X) .The process of tree blinding is depicted in figure 5.

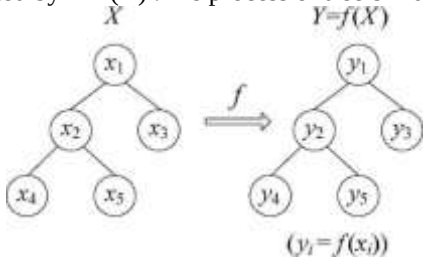

Fig.5: Tree blinding

One-way functioning of 'f' helps to hide information about each secret keys of a key tree without compromising its structure.

There are 3 algorithms given by Jing and Yang in [6] .

1) Adding Multiple Leaf Nodes.
2) Removing Multiple Leaf Nodes.
3) Changing the Values Of Leaf Nodes.

There are some concepts in those algorithms such as Incremental tree, Combined Ancestor tree, Ancestor chain, Normalization and Expansion.

Incremental trees are built using methods that allow an existing tree to be updated or revised using new and individual data. This is useful in several situations: (a) the entire data is not available at the time the original tree is built,

(b) the original data is too large to process, or (c) the data change's characteristics over time.

For a set of leaf nodes, the subtree consisting of all ancestors of those leaf nodes is called a CAT(Combined Ancestor Tree) [4]. An ancestor chain is an instance of a CAT that has one single leaf node.

The normalizing tree is constructed from CAT by first changing the value of each leaf node secret key of CAT into its corresponding inverse in G , and then computing all the other internal node secret keys.

III.NECESSARY CONDITION

Collusion between any evicted member and joining member is useful for calculating GK and knowing the Previous and future messages is possible only when following conditions hold:

(i) If Evicted member and joining member are at different subtrees.
(ii) Group Rekeying is not happened after eviction (atleast before new member joining)
(iii) Users can access the group information with previous group keys.

So We are proposing a scheme named OFT with SGRK(Secure Group ReKeying) in the following section to resist the collusion according to the above conditions.

IV.PROPOSED SCHEME

There are several tree based MKD protocols such as LKH (logical key hierarchy), OFT (one-way function tree) and HOFT (homomorphic one-way function tree). In these protocols, to update the group key in the event of member eviction or joining, various group rekeying schemes have been proposed. But they are not optimal because of their computational overhead. And OFT has another problem that it is vulnerable to collusion attack. Group rekeying in HOFT requires normalization, contraction and expansion steps. These steps are complex to implement. So a new scheme namely OFT with SGRK (Secure Group Rekeying) is proposed. In OFT, MD5 hashing was used which was not secure according to CMU Software Engineering Institute. So SHA-2 family hash function SHA-256 is used. Before Going into scheme details, let us go through following key words .

Collusion:

It is an illegal co-operation between group members(newly joined and evicted) in order to obtain group key and to know the previous messages.

One-way Function:

Function that " on every input computes easily , but hard to invert with the given random input".

PRNG(Pseudo Random Number Generation):

An algorithm that is used to produce an open-ended sequence of numbers is referred to as a PRNG.The sequence of random numbers{ Xn } is obtained via the following iterative equation :

$Xn+1 = (a\ Xn + c) \bmod m$

m is modulas ,a is multiplier , c is increment and Xn is starting number.

If m, a ,c and Xn are integers, then this technique will produce a sequence of integers with each integer in the range $0 \leq Xn < m$.

BigInteger :

Here we are generating 256-bit numbers for security purpose. For generating 256 bit integers, BigInteger concept is used. BigIntegers are Immutable arbitrary-precision integers. All operations on BigIntegers behave as if BigIntegers were represented in two's-complement notation (like Java's primitive integer types).

SHA-256:

SHA-2 is the successor of SHA-1.SHA-256 should be chosen in critical applications where a high speed hash function is needed. SHA-256 is considered secure with no known theoretical vulnerabilities.

In OFT with SGRK , collusion resistance is achieved by doing these following 3 steps:

1.Key Generation
2.Key Assignment
3.Group Rekeying

Key Generation:

In order to generate the keys, PRNG and SHA-256 were used. Generate a 256-bit random number (by PRNG), each as the secret key for each member. Blinding function (SHA-256) is applied to the generated SK for calculating the BSK (blinded secret key). IBSK (internal node blinded secret key) is a result of XOR operation between two BSKs of leaf nodes. GK is calculated by doing XOR operation between two root-nearest IBSKs.

Key Assignment:

Own Blinded secret key is directly supplied to the group members. Group server sends all the needed Blinded secret keys of other members in encrypted form. Now, group key can be calculated. In this project, symmetric encryption scheme DES (Data Encryption Standard) is used for encryption and

decryption. For example, group member will be supplied with "asgrhdufgkrigk (cipher text)" instead of 1453759753(plain text).

Group Rekeying:

Preserving the Group forward and backward secrecy is indespensable in a Group key distribution scenario. For that, we are reassigning the keys when a member joins/leaves the Group.

Member Addition

When a new member joins the group, an existing leaf node x that is closest to the root is split, creating new nodes left(x) and right(x). The member associated with x becomes left(x) while the new member becomes right(x) and is given a new secret key. Then, the new values of the changed blinded secret keys are securely broadcast to the appropriate subgroups according to the blinded secret key updating operation .An example of adding a member is shown in Figure 6.

Member Eviction

When the member associated with leaf node *y* is evicted from the group, node *y*'s sibling, denoted by *s*, is reassigned to the parent of *y*. If *s* is a leaf node, its associated member is given a new secret key. Otherwise, i.e., *s* is

The root of a sub tree, one leaf node of this sub tree is given a new key. Then, the new values of the changed blinded secret keys are securely broadcast to the appropriate subgroups according to the blinded secret key updating operation. An example of evicting a member is shown in Fig.7.

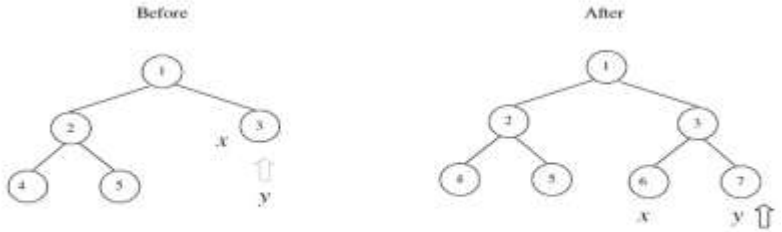

Figure 6: Adding a member

Let x=SK, y=BSK, z=GK, f=SHA-256(Blinding function) , g=XOR(mixing function).

Algorithm for Preventing Collusion attack

Step 1: Genretae a 256-bit random number 'x' using PRNG.

Step 2: Now apply Blinding function 'f' to x, we get y.

y=f(x)=SHA-256(x).

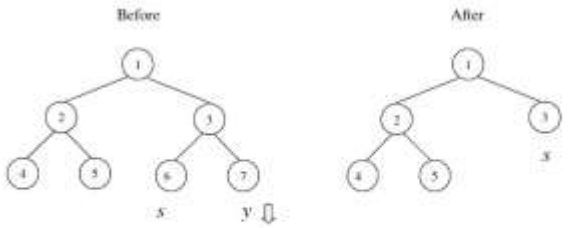

Figure 7: Evicting a member

Step 3: Group members are registered to the group through secure user name-password scenario.

Step 4: Group members are supplied with their own BSK in original and other BSK's needed by that member encrypted with DES by that member's BSK.

Step 5: y is Group key (When there is only one member in the group)

Step 6: Now Group member uses BSKs to know the group key.

Step 7: If Group members are more than one then IBSK(internal node Blinded secret key) is calculated. IBSK=g(BSK1,BSK2)

Step 8: GK is calculated as: GK=g(IBSK1,IBSK2) (i.e. XOR operation between two root-nearest IBSKs)

Step 9: After every member joining/leaving , GS sends the updated keys to the members who need them to update their GK.

Step 10: Group rekeying is done after every member's joining/leaving. So an evicted member and joining member's collusion is insufficient to know the GK.

Step 11: Collusion resistance is the resultant of step 10.

V. PERFORMANCE ANALYSIS

In OFT, MD5 hashing function was used. In OFT with SGRK, SHA-256 was used. SHA-256 is very faster when compared to MD5 and other SHA algorithms as depicted in the following table 2.

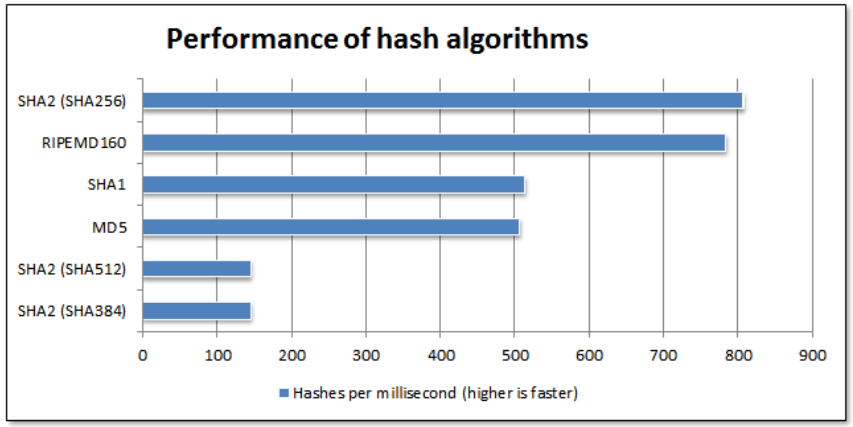

Table 2: Performance comparison of hash algorithms

In below table 3, comparison of OFT, HOFT and OFT with SGRK is given.

Scheme name	One-way function used	Mixing function used	Collusion attack	Computational overhead
One-way function tree (OFT)	MD5 Hash	XOR	YES	Moderate
Homomorphic one-way function tree (HOFT)	Rabin function	Modular multiplication	NO	High
OFT with SGRK	SHA-256	XOR	NO	Low

Table 3: Comparison of OFT, HOFT and OFT with SGRK

VI.CONCLUSION

OFT is vulnerable to collusion attack. By using OFT with SGRK, a collusion free multicast key distribution scheme has been proposed. OFT with SGRK is quite simple in terms of computational overhead when compared to previous tree-based MKD protocols. Developing rigorous analysis and automatic verification methodology for this protocol is necessary. There is no research result related to formal verification of OFT protocol.

VII.FUTURE WORK

The future work can focus on developing a formal verification of OFT with SGRK protocol and also using 512 and 1024 bit keys with less computational and communicational overhead.

VIII. REFERENCES

[1] L. Cheung, J. A. Cooley, R. Khazan, and C. Newport, "Collusion-resistant group key management using attribute-based encryption," in Proc.1st Int. Workshop Group-Oriented Cryptographic Protocols (GOCP), 2007.

[2] C. K.Wong,M. Gouda, and S. S. Lam, "Secure group communications using key graphs," IEEE/ACM Trans. Netw., vol. 8, no. 1, pp. 16–30, Feb. 2000.

[3] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key management for multicast: Issues and architectures," Internet Draft, Internet Eng. Task Force, 1998.

[4] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," IEEE Trans. Softw. Eng., vol. 29, no. 5, pp. 444–458, May 2003.

[5] D. Balenson, D. McGrew, and A. Sherman, "Key management for large dynamic groups: Oneway function trees and amortized initialization," draft-irtf-smug-groupkeymgmt-oft-00.txt, Internet Research Task Force, August 2000.

[6] J. Liu, and B. 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, 2011.

[7] R. C. Merkle, Secrecy, Authentication, and Public-Key Cryptosystems, Technical Report No.1979-1, Information Systems Laboratory, Stanford University Palo Alto, Calif, 1979.

[8] G.Horng, "Cryptanalysis of a keymanagement scheme for securemulticast communications," IEICE Trans. Commun., vol. E85-B, no. 5, pp. 1050–1051, 2002.
[9] W. C. Ku and S.M. Chen, "An improved key management scheme for large dynamic groups using one-way function trees," in Proc. Int. Conf.
Parallel Processing Workshops, 2003, pp. 391–396.
[10] X. Xu, L.Wang, A. Youssef, and B. Zhu, "Preventing collusion attacks on the One-Way Function Tree (OFT) scheme," in Proc. 5th Int. Conf. Applied Cryptography and Network Security, Zhuhai, China, 2007, pp.177–193.
[11] M. O. Rabin, Digitalized Signatures and Public-Key Functions as
Intractable as Factorization. Cambridge, MA, Massachusetts Inst.
Technol., Lab. Comput. Sci., 1979.
[12] Jing Liu, Qiong Huang, Bo Yang, and Yang Zhang, "Efficient Multicast Key Distribution Using HOWP-based Dynamic Group Access Structures ", Manuscript received 2011.

P.Aravind received master of technology degree in computer science and engineering from JNTU university, hyderabad. He has been in teaching profession since 2004. He is working as Associate professor in Gayatri vidya parishad college of engineering,visakhapatnam. He is advisor in Institute of Engineers,India for students section. He is having research interest in cryptography, network security and its applications.

Syed.Abbas pursuing master of technology degree in computer science and engineering from Gayatri vidya parishad college of engineering affiliated to JNTU university, kakinada . He is having research interest in cryptography, network security and its applications.