Global Journal of Computer Science and Technology
Network, Web & Security
Volume 13 Issue 11 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Group Key Management Techniques

By Mridula R. D. & Sreeja Rajesh
SCMS School of Engineering & Technology (SSET), India

Abstract - The most widely used technique in a network is Group communication. This helps in the reduction of the bandwidth usage. The major concern in group communication is its security of messages. Group key provides security of messages and hence proper group key management is very important in a group communication. There are various classifications of group key management techniques. A survey of these key management techniques is done in this paper.

Keywords : *group communication, group key management.*

GJCST-E Classification : *E.3*

Strictly as per the compliance and regulations of:

Group Key Management Techniques

Mridula R. D. [α] & Sreeja Rajesh [σ]

Abstract - **The most widely used technique in a network is Group communication. This helps in the reduction of the bandwidth usage. The major concern in group communication is its security of messages. Group key provides security of messages and hence proper group key management is very important in a group communication. There are various classifications of group key management techniques. A survey of these key management techniques is done in this paper.**

Keywords : group communication, group key management.

I. Introduction

The most widely used technique in a network is group communication. Group communication is used in group chat, video /audio conferencing, sending software updates, dividing /sharing work among a group in a corporate environment, multi-party gaming, teleconferencing, telemedicine etc. Security, bandwidth management, speed etc are the various concerns on group communication. If the communication is properly designed and managed, then it will help in the effective usage of band width. The most critical problem that has to be addressed in any group communication is the security of its messages. Group key management is the most important among all its security problems.

Multicast is an efficient technology that supports group communication. It helps in better utilization of network resources. Group key needs to be shared among all the members, to ensure security in group communication and also it needs to be maintained secure and fresh. This helps to ensure that only authorized users have group key. Every messages has to be encrypted with group key before transmitting. Thus outsiders or intruders are unable to interpret the messages even though they receive the encrypted message.

In any practical application, the network has to be scalable and dynamic. Frequent membership changes might be there in such networks. With every membership change, key management operation has to be performed to ensure that it follows the four main rules in key management backward security (a new member joins the group should not have access to any of its past messages), forward security (a member who have left the group should not have access to its future information packets), collusion freedom (deleted members should not be able to deduce the group keys) and group confidentiality (users that were not part of the group ever in the past should not have access to any key in the multicast group).

Proper group key management is critical for secure group communication. Various classifications on group key management techniques are discussed in next section.

II. Classifications of Group Key Management Protocols

Based on 'how' the key management operations are performed, the protocols are classified into centralized, de-centralized, and distributed/ contributory. In centralized group key management protocols there is a central group key server, which will be completely responsible for updating and distributing the group keys. Though this method is simple, the existence of single key server generates a bottleneck in the system. In de-centralized group key management systems, the entire group is divided into distinct subgroups and each group has a sub group controller. This sub group controller is responsible for key management operations in sub group. Also at the time of message transmission, this performs message relaying operations and so introduces delays in message transmission. In contributory/ distributed group key management each group member has an equal share to contribute to the group key. This avoids the problem with centralized trust and single point of failure.

Depending on 'when' the group key is updated, key management techniques are divided into three: time driven, message driven and membership driven. Group key is updated at regular time intervals, for time driven techniques. This helps to reduce the number of rekeying operations in highly versatile group and also ensure security of the system. In message driven key management protocols, the rekeying happens along with each transmitted message. This helps to ensure the forward and backward security. In membership driven group key management protocols, the group key is updated when a member joins or leaves a group.

In the rest of this paper we focus more on the first category of protocols. Some examples of centralized and decentralized group key management protocols are discussed in the next section. In the rest of the sections we concentrate more on various distributed

Author α : Senior Systems Engineer, Infosys Limited, Trivandrum, Kerala. E-mail : mridulard@gmail.com
Author σ : Department of Computer Science & Engg, SCMS School of Engineering & Technology (SSET), Karukutty, Kerala.
E-mail : m.sreeja79@gmail.com

key management techniques and their performance analysis.

III. Centralized Protocols

As discussed in the introduction section, there will one central key server in centralized techniques. This key server will be responsible for the whole re-keying process. Each member has a shared key called Key Encryption Key (KEK) with the key server. Thus for an n-member group there will be n-keys and the server maintains a list of group members and keys. Anytime when server generates new group key, it encrypts the new group key with n KEKs and send those packets to corresponding group member. Each member then decrypts the packets using their KEK and retrieves the group key. Thus every member receives the same new group key. Every time when a member joins or leaves a group, the key server generates and distributes new group key to ensure forward and backward security. In case of large dynamic group, it will be a serious burden on key server to generate, encrypt and distribute n keys in short time. Transmission of n encrypted packets greatly increases the bandwidth usage. Some of the centralized key management techniques are described below.

a) GKMP

Group Key Management Protocol (GKMP)[1] is proposed by Harney and Muckenhirn[3]. This is a member driven protocol. The secret key (KEK) is shared between server and each member. In this method the server generates a group key packet (GKP) which contains a group traffic encryption key (GTEK) and a group key encryption key (GKEK). When a new member joins a group, the server generates new GKP and sends it securely to the new member by encrypting it with the KEK established with new member. With existing members it sends the new GKP by encrypting it with old GTEK. When a member leaves, the server generates new GKP and distributes it to the remaining members by encrypting it with KEK shared with each member. This ensures forward and backward security. But this method requires O(n) messages for each re-keying and so this method is not suitable for large dynamic groups.

b) Hao-Hua Chu's Protocol

This method is proposed by Hao-Hua Chu et al[2]. This is a message driven protocol. When a member wants to multicast a message, it generates new TEK and encrypts the message before transmitting. It also sends the TEK to group server encrypting it with the KEK shared between the member and group. Server decrypts the TEK using KEK and then the server unicasts the TEK to remaining group members by encrypting each message with the KEK shared between corresponding member and the server. The members then decrypts the message from server and retrieves the new TEK and then uses this key to decrypt the message from the initial group member. Also with every membership change the key server generates new TEK and distributes it to each member. But this adds the burden on server.

c) LHK Protocol

This is a membership driven protocol. The basis of this method[3] is the logical hierarchical key tree structure. This tree structure will be maintained in server. Root of the key tree is the group key. Leaf node contains the secret key shared between server and individual user. Intermediate keys are used in the distribution of new group keys. Out of all these keys, each member uses only the keys that lie on the path from that user till the server. So along with each membership change, the keys in the affected path has to be updated and re-distributed. When a member joins or leaves a group, the key server generates new group key and intermediate keys in the affected path. Then it securely distributes the keys to the corresponding group members. This method is more scalable compared to other unicast based approaches. For a group of N members with degree of key tree as d, the communication cost will be O(log(dN)). But for the above mentioned unicast approaches it is O(n). Since this is also a centralized method all the disadvantages of centralized methods will be there for this method also.

d) Code for Key Calculation (CKC)

This protocol[4] is proposed by M. Hajyvahabzadeh, E. Eidkhani, S. A. Mortazavi and A. Nemaney Pour. This method is also based on logical key hierarchy. Unlike LHK, the intermediate node keys are calculated by individual users. When a member joins or leaves a group, the server sends only group key to the members. By using this key the members calculate other keys using node codes and a one way hash function. The security of this method is based mainly on the one wayness/strength of hash function. By this method it reduces the server overhead and also the message size.

There are some more works in this category of group key management protocols like Secure Lock[5], One-way Function Tree[6], Centralized Flat Table Key Management[7] etc.

IV. Decentralized Protocols

In decentralized techniques, the entire group is divided into several subgroups. Group key is shared among all the members and each sub group has sub-group key shared among the members of that sub group. There will be one central key server and a subgroup key server for each subgroup. Some examples of this method is described below:

a) IOLUS

Iolus [8] is proposed by S. Mittra. In this method is based on a secure distribution tree, in which all the

members are divided into certain sub-groups and these sub groups are arranged hierarchically to form a virtual secure group. When a user wants to join a multicast group, it locates its designated GSA (Group Security Agents) and sends a JOIN securely. On receipt of that request the GSA decides whether to approve or deny the request. When request is approved, it generates a secret key shared between new member and GSA and it communicates the key securely to the new member. GSA then saves all the relevant details about the new member in its secure private data base. It then sends out a GROUP KEY UPDATE message securely to all the existing members. This message contains the new sub group key encrypted with old sub group key and it also securely communicates to the new member the sub group key through a secure channel.

b) KRONOS

Setia et al[9] proposed this scalable approach. This is a time-driven approach and thus frequency of rekeying is independent of the group size and its dynamicity. Kronos is built on the key management framework IGKMP. The working is also similar to IGKMP with a major difference that Kronos is period based re-keying technique.

Some other examples of decentralized group key management techniques are Hydra[10], Safecast and MARKS[11]. The main drawback with these methods is that, long-term secure channels needs to be established by the key server with all the group members. This increases the cost of introducing new key server.

V. Distributed Group Key Management Protocols

Various distributed key management techniques like (DHSA, EDKAS, TGDH, DGKD), will be discussed in this section. All the four are membership driven protocols and so the major two operations which requires attention is member join and member leave. Member join and leave operations for all the above four techniques are discussed below.

a) EDKAS (A Efficient Distributed Key Agreement Scheme using one Way Function Trees)

This method[12] is based on the concept of distributed one way function trees. This is a period based group rekeying approach. This method takes an assumption that, all the members has already been passed through some admission control methods to make it authentic.

In this method, each leaf node is assigned one ID and with root node ID as 0. For any non-leaf node with ID v, its child nodes will have IDs (2v+1) and (2v+2). Each leaf node represents the members. Each member has its own secret key and blinded key (generated by applying one way hash function). The secret key of a node can be calculated from the blinded keys of its child nodes, using a mixing function ($K_v = f(B_{K2v+1}, B_{K2v+2})$). In this way the secret key associated with the root node (known as group key) is shared by all the members. Each member holds its own secret key. It also holds all the blinded keys of nodes that are sibling of the nodes in its key path starting from its associated leaf node up to the root node of the tree. A responsible member set, RM, is also associated with a node, which contains members in the sub tree rooted at its sibling node.

Member join operation is explained in Fig 1. U_7 wants to join the group. 6 is the insertion node and U_5 is the sponsor. Blinded key BK_{14} of U_7 is send to U_5. U_5 regenerates its secret key K_{13} and its blinded key BK_{13}, BK_6 and BK_2. U5 then sends BK_6 to U_4, BK_2 to U_1,U_2 and U_3. It also sends the structure of distributed one way function tree structure, BK_{13}, BK_5 and BK_1 to U_7. Now at this step all the members have the required information to generate group key K_0. The member leaving case is similar to that of join, with sibling node as the sponsor and this node is promoted to leaving nodes parent position. Then as discussed above, the sponsor initiates the re-keying operations Fig 2.

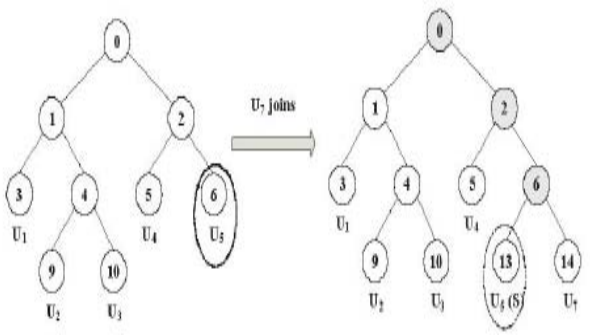

Figure 1 : EDKAS Join Operation

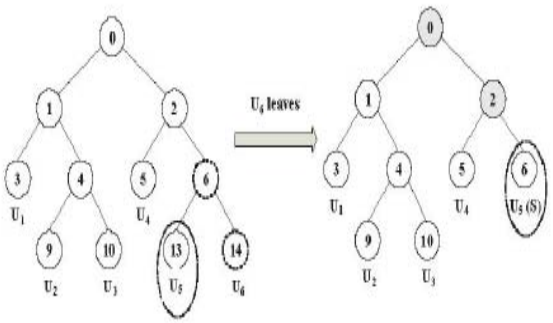

Figure 2 : EDKAS Leave Operation

This is actually a period based method. So the above single node join case is extended to a batch join and so upon each join a temporary key tree structure is generated and kept aside. At the beginning of each period, the temporary tree is merged to the actual tree structure.

Since this method is period-based, it decouples the frequency of rekeying from the size and membership

dynamics of the group. Therefore, this scheme can easily scale to dynamic collaborative groups. Though this method is theoretically efficient, its practical implementation is expensive.

b) TGDH (Tree based group key agreement scheme) [13]

The concept of hierarchical key tree and multi-party Diffie-Hellman is used in this method. The leaves of the key tree represent users.

In this method new node join requires two rounds of operation. A new node broadcasts a join request containing its own blinded key. The blinded key is calculated by applying modular exponentiation operation on its secret key. Upon receipt of this message, each node calculates the insertion position. New node will be inserted to the shallowest point in the tree, so that it does not increase the tree height. Sponsor will be the right most leaf rooted at the insertion node. Each member creates a new intermediate node with new node and sponsor as its children. After this step, all the members will be blocked except sponsor node. The sponsor generates new secret key and calculates its blinded keys. Since it contains the blinded keys of all the other nodes, it can calculate the new group key. Then sponsor broadcasts all the blinded keys. Then all the other members and the new member can calculate the new group key.

The leave protocol is similar to that of join. The sponsor is the rightmost leaf node of the sub tree rooted at leaving nods's sibling. All the members update their tree structure by deleting the leaving node and promoting the sibling node of leaving node to the parent position of leaving node. Similar to that of join, the sponsor re-calculates new key and the blinded keys and broadcasts it to other members. The members then can calculate the new group key.

Since this protocol requires rekey initiation after each membership change, the cost of modular exponentiation makes the entire system slow.

c) DGKD (Distributed Group Key Distribution) [14]

The concept of sponsor and co-distributer is used in this method. This method is based on hierarchical tree structure. At join/leave, the sponsor generates new group key and initiates key distribution operation. The sponsor distributes new key with the help of co-distributers. Since this is distributed method all the group members are equally capable and mutually trusted. Depending on the relative location of joining/leaving member, any group member can have the potential sponsor.

Every member has a sponsor field which will be updated, if it is along the joining member's path. If new members sponsor id is greater than that of the node's sponsor id, then the sponsor id is replaced with the new node's id. In this method, the co-distributor is responsible for generating the affected intermediate node keys. The sponsor might not be having the keys along other branches, co-distributor helps in distributing keys to other individual nodes.

The new node, m_{n+1}, makes a join request by broadcasting its public key PK to all existing members $m_1,\ldots,m_n$. The right most member replies to this node after authenticating it. It decides and broadcasts the insertion location of new node. It then sends the virtual key tree and the list of public keys of other nodes to the new member. Then the sponsor member is decided. The new node's sibling node becomes the sponsor. If there is no sibling node, the new node itself becomes its sponsor. The sponsor node generates and distributes the new keys along its path till root. If requires members update the sponsor id also. In a group like the one shown in Fig 3, m4 generates new keys k'_{4-5}, k'_{4-7} and k'_{0-7} and broadcasts the encrypted keys using co-distributers public keys like, $\{k_{4-7}, k_{0-7}\}$ Pk_7 and $\{k_{0-7}\}$ Pk_3. Co-distributers will decrypt the keys and then decrypt using intermediate node keys and then broadcast the messages to other members. The messages will be $\{k_{0-7}\}$ k_{0-3} by m_3 and m_7 messages will be $\{k_{4-7}\}$ k_{6-7} and $\{k'_{0-7}\}$ k_{4-7}. m_4 also encrypts and sends the key to m_5: $\{k'_{4-5}, k'_{4-7}, k'_{0-7}\}$ Pk_5.

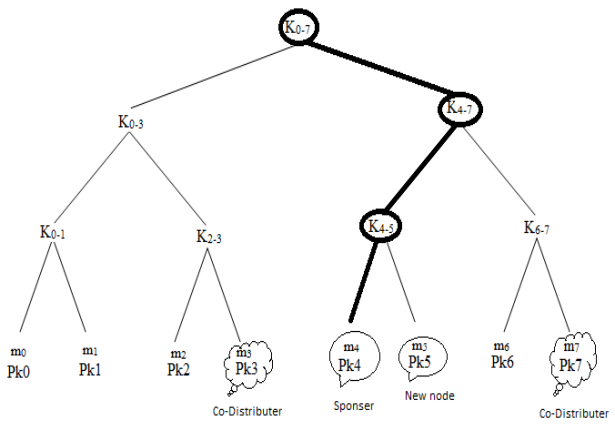

Figure 3 : DGKD Join Operation

In member leave operation, sibling will act as sponsor (For m_5 m_4 will be the sponsor). m_4 generates the new keys, k'_{4-5}, k'_{4-7} and k'_{0-7}. m_4 broadcasts the encrypted keys using co-distributers public keys like, $\{k_{4-7}, k_{0-7}\}$ Pk_7 and $\{k_{0-7}\}$ Pk_3. Co-distributers will decrypt the keys and then decrypt using intermediate node keys and then broadcast the messages to other members. The messages will be $\{k_{0-7}\}$ k_{0-3} by m_3 and m_7 messages will be $\{k_{4-7}\}$ k_{6-7} and $\{k'_{0-7}\}$ k_{4-7}. Thus all the members will get new keys (Fig 4).

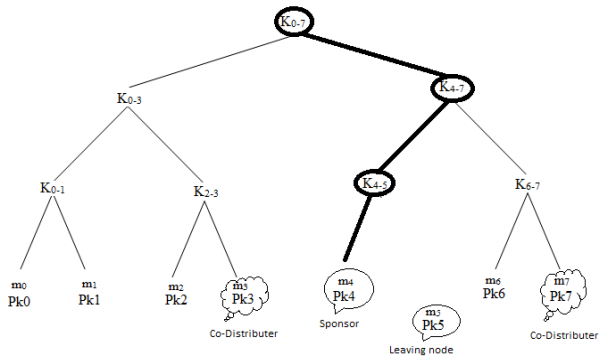

Figure 4 : DGKD Leave Operation

This method it uses special authentication methodologies. When m4 transmits new keys to m_3, the packet contains two components. One is k_{0-7} signed using m_4 private key and k_{0-7}. So that m3 can decrypt and verify the authenticity of message. m_3 while transmitting the message to other members, it keeps the signed k_{0-7} also so that each member can verify that the message originally came from m_4.

There are mainly two drawbacks for this method. All the affected intermediate keys have to be generated by the sponsor member, which will increase the work load of sponsor. Also this method uses asymmetric cryptosystem, which is slower than symmetric system.

d) DHSA (Distributed Group Key Management using Hierarchical Approach with Diffie-Hellman and Symmetric Algorithm) [16]

As name indicates, this distributed group key management approach uses Diffie-Hellman and symmetric algorithm along with the concept of logical hierarchical key tree. In the key tree structure, the public key of each member is stored in leaves and the intermediate nodes contain the symmetric keys. Two types of codes are used in this method – binary code and decimal code. Binary code is used for identifying the position of a member and decimal code is used in the calculation of intermediate node keys. A list containing public key of all the members and their binary codes (called member list) is shared by all the group members. On each membership change this list will be updated. Root node will contain the group key. Intermediate node key is calculated using the below formulae.

$$Key_{intermediate\_node} = f(Key_{group} \ XOR \ Code_{intermediate\_node}).$$

$$Code_{child\_node} = (Code_{parent\_node} \ || \ Random \ digit).$$

A sample hierarchical key tree structure is shown in Fig 5. When a new member wants to join a group he/she sends a join request message to the entire group. The node with no siblings will reply .If there are multiple nodes having no siblings, then the node with smallest parent binary code value replies to the join request. On receipt of this join request each member check if it has the smallest binary code value, if so then that node will be responsible for the key management operations at this join. Consider a group with 7 members and joining node U4 (Fig 6). U4 broadcasts a join request to all the seven members. U3 does not have a sibling node so U3 will act as sponsor for U4. It authenticates U4. Both U3 and U4 exchanges the public keys and establishes a shared key (g^{X3X4} mod p, where X3 is the private key for U3 and X4 is the private key for U4.) using Diffie-Hellman key agreement scheme. U3 adjusts its position to accommodate U4. U3 also calculates the intermediate node codes and key for new node. The updated binary code for U3and new position and public key for U4 are inserted into member list table. At this moment all the other

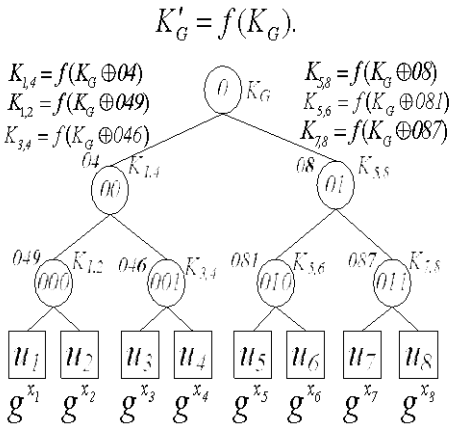

Figure 5 : DHSA Hierarchical key tree structure

nodes calculates new group key by taking hash value of existing group key. U3 then encrypts the new group key using the Diffie-hellman shared key and send it to the new member U4. Then the members in the affected path will calculate the intermediate node keys using the decimal code and new group key.

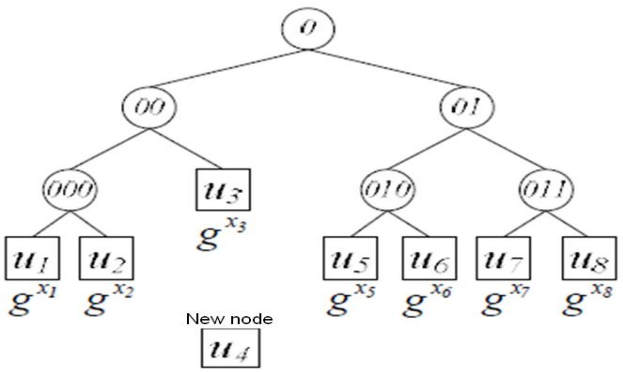

Figure 6 : DHSA Join

When a member wants to leave a message, then its sponsor will be its sibling node. All the entries, corresponding to the leaving member will be deleted from the shared member list table and the sibling member adjusts its position upwards in the key tree and

this new parent binary code will also be updated in the member list table. At this moment all the other members stops its transmissions for a while and listens to the sponsor (sibling node) for new group key. Now the sponsor node calculates the new group key by applying the symmetric algorithm, one time pad. To reduce the key packet transmission the group key is transmitted in a specific order (as shown in Fig 7). The entire group members are divided into (log n -1) groups ({U1, U2, U3, U4}, {U5, U6}) and one member from each group is randomly selected (say U1 and U5). The sponsor member then uncast the group key to those nodes by encrypting with their shared keys. Then the representative members (U1 and U5) will multicasts the group key to other members by encrypting the new group key with their common intermediate node's (nodes $U_{1\text{-}4}$ and $U_{5\text{-}6}$ here) key.

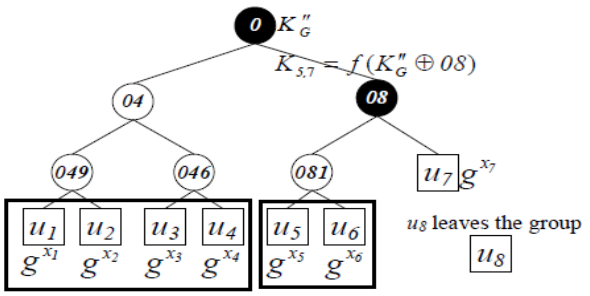

Figure 7 : DHSA Member Leave

The advantage of this method falls in join operation rekeying. In join operation, the group key is transmitted only once in one message i.e. between new node and sponsor.

e) Analysis

We discussed four different distributed key management approaches here. Their performance analysis based on key generation overhead and key communication overhead are discussed here. Key generation overhead is the number of keys generated by the sponsor member. The number of messages required to transmit the group key is key communication overhead.

The key generation overhead for DGKD and EDKAS are almost similar. The key generation over head is least and constant for DHSA. Because, for DHSA the sponsor node generates only one group key. All the other nodes calculate the group key by taking hash value of existing.

Node Count	Number of keys Generated					
	Member Join			Member Leave		
	EDKAS	DGKD	DHSA	EDKAS	DGKD	DHSA
6	4	3	1	2	2	1
7	4	3	1	2	2	1
10	6	4	1	4	3	1
11	6	4	1	4	3	1
12	6	4	1	4	3	1
23	8	5	1	6	4	1
24	8	5	1	6	4	1
25	8	5	1	6	4	1
27	8	5	1	6	4	1
28	8	5	1	6	4	1
30	8	5	1	6	4	1

Table 1 : Key generation overhead analysis for join and leave operations

For join operation, DHSA has communication overhead 1. Because, the sponsor transmits group key only to the new member. There is no group key exchange between existing members and sponsor. Communication overhead is the highest for EDKAS, because sponsor sends the keys individually to each member. At member leave, the communication overhead is the same for DGKD and DHSA. But the message size of DHSA is the least, since it contains only group key.

Node Count	Number of messages send					
	Member Join			Member Leave		
	EDKAS	DGKD	DHSA	EDKAS	DGKD	DHSA
6	5	3	1	4	2	2
7	6	3	1	5	2	2
8	7	5	1	6	4	2
9	8	5	1	7	4	4
10	9	5	1	8	4	4
24	23	7	1	22	6	6
25	24	7	1	23	6	6
27	26	7	1	25	6	6
28	27	7	1	26	6	6

Table 2 : Key communication overhead analysis for join and leave operations

VI. Conclusion

Various classifications of group key management techniques are discussed in this paper. We concentrated more on four different distributed key management techniques such as EDKAS, TGDH, DGKD and DHSA. From the performance analysis of the four methods, it is clear that, for new member join case, DHSA has the least key generation, key encryption and communication overheads and is a constant indicating that DHSA is more scalable than other methods.

References Références Referencias

1. H. Harney and C. Muckenhirn. "Group Key Management Protocol (GKMP) Architecture". July 1997, RFC 2093.
2. H. Chu, L. Qiao, K. Nahrstedt. "A Secure multicast Protocol with Copyright Protection". ACM SIGCOMM Computer Communications Review, April 2002.
3. D. Wallne, E. Harder, and R. Agee "Key Management for Multicast: Issues and Architectures," National Security Agency, RFC2627, 1999.
4. M. Hajyvahabzadeh, E. Eidkhani, S. A. Mortazavi and A. Nemaney Pour, "A New Group Key Management Protocol using Code for Key Calculation: CKC," Proceedings of IEEE, IEEE Computer Society, the International Conference on Information Science and Applications (ICISA2010), Seoul, Korea, pp. 1-6, Apr. 2010.
5. G. H. Chiou and W. T. Chen. Secure broadcast using the secure lock. IEEE Transactions on Software Engineering, 15(8): 929–934, August 1989.
6. D.A. McGrew and A.T. Sherman, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,.
7. M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. "The Varsakey Framework: Versatile Group Key management". IEEE Journal on Selected Areas in Communications (Special Issues on Middleware), 17(8): 1614-1631, August 1999.
8. S. Mittra, Iolus: "A Framework for Scalable Secure Multicasting," Proc. of ACM SIGCOMM'97, pp. 277–288, Oct. 1997, doi: 10/1/1/39/4261.
9. S. Setia, S. Koussih, S. Jaodia, and E. Harder. "Kronos: A scalable Group Re-keying Approach for Secure Multicast". Proc. of IEEE Symposium on Security and Privacy, 2000.
10. Rafaeli & D. Huchison, "Hydra: a decentralized group key management," Proc. of the 11th IEEE International WETICE: Enterprise Security Workshop, pp. 62–67, Nov. 2002.
11. B. Briscoe. "Marks: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences" Proc. First International Workshop on Networked Group Communication (NGC), Pisa, Italy, November 1999.
12. J. Zhang, V. Li, C. Chen, P. Tao and S. Yang, "EDKAS: An Efficient Distributed Key Agreement Scheme Using One Way Function Trees for Dynamic Collaborative Groups," IMACS Multi-conference on CESA, Bejing, China, pp. 1215-1222, Oct. 2006, doi: 10.1109/CESA.2006.313506.
13. Fan, L. Xiao-ping, D. Qing-kuan and L. Yan-ming, "A Dynamic Layering Scheme of Multicast Key Management," IEEE 5th International Conference on Information Assurance and Security (IAS '09), Xian, China, pp. 269-272, Aug. 2009, doi: 10.1109/IAS.2009.344.
14. P. Adusumilli, X. Zou and Byrav. R "DGKD: Distributed Group Key Distribution with Authentication Capability" Proc. of the 2005 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, pp. 693-698, Jun. 2005, doi: 10.1109/ISCC.2000.860720.
15. HS. Anahita Mortazavi, Alireza Nemaney Pour, Toshihiko Kato "An Efficient Distributed Group Key Management using Hierarchical Approach with Diffie-Hellman and Symmetric Algorithm" FEB 2011.

This page is intentionally left blank