International Journal of Engineering Inventions
ISSN: 2278-7461, www.ijeijournal.com
Volume 1, Issue 3 (September 2012) PP: 27-31

A Dynamic Approach to Multicast Communications Using Decentralized Key Management

D.Chalapathirao[1], K.Rajendra Prasad[2] , S.Venkata Rao[3]
[1]Department of CSE, Chaithanya Engg.College, Visakhapatnam, A.P., India
[2]Assoc. Professor, Department of CSE, Chaithanya Engg College, Visakhapatnam, A.P., India.
[3]Assist. Professor, Department of CSE, Kaushik College of Engg, Visakhapatnam, A.P., India.

***Abstract*— The security in the multicast communication in the large groups is the major obstacles for effectively controlling access to the transmitting data. The IP Multicast itself does not provide any specific mechanisms to control the intruders in the group communication. Group key management is mainly addresses upon the trust model developed by Group Key Management Protocol (GKMP). There are several group key management protocols that are proposed, this paper will however elaborate mainly on Group key management which has a sound scalability when compared with other central key management systems. This paper emphases protocol which provides a scope for the dynamic group operations like join the group, leave the group, merge without the need of central mechanisms. An important component for protecting group secrecy is re-keying. With the combination of strong public and private key algorithms this would become a better serve to the multicast security.**

***Index Terms*—Group key management, multicast security, scalability.**

I. INTRODUCTION

1.1 Unicast - Broadcast Multicast

The multicast group can be identified with the class D IP address so that the members can enter or leave the group with the management of Internet group management protocol. The trusted model gives a scope between the entities in a multicast security system. For secure group communication in the multicast network, a group key shared by all group members is required. This group key should be updated when there are membership changes in the group, such as when a new member joins or a current member leaves. Along with these considerations, we take the help relatively prime numbers and their enhancements that play a vital role in the construction of keys that enhances the strength for the security.

Multicast cryptosystems are preferably for sending the messages to a specific group of members in the multicast group. Unicast is for one recipient to transfer the message and ‘Broadcast’ is to send the message to all the members in the network. Multicast applications have a vital role in enlarging and inflating of the Internet. The Internet has experienced explosive growth in last two decades. The number of the Internet users, hosts and networks triples approximately every two years. Also Internet traffic is doubling every three months partly because of the increased users, but also because of the introduction of new multicast applications in the real world such as video conferencing, games, ATM applications etc.. broad casting such as www, multimedia conference and e-commerce, VOD (Video on Demand), Internet

Broadcasting and video conferencing require a flexible multicasting capability. Multicast is a relatively new form of communications where a single packet is transmitted to more than one receivers. The Internet does not manage the multicast group membership tightly. A multicast message is sent from a source to a group of destination hosts. A source sends a packet to a multicast group specifying as the multicast group address. The packet is automatically duplicated at intermediate routers and any hosts that joined the group can receive a copy of the packet. Because a host can receive transmitted data of any multicast groups, secure communications is more important in multicasting than in unicasting.

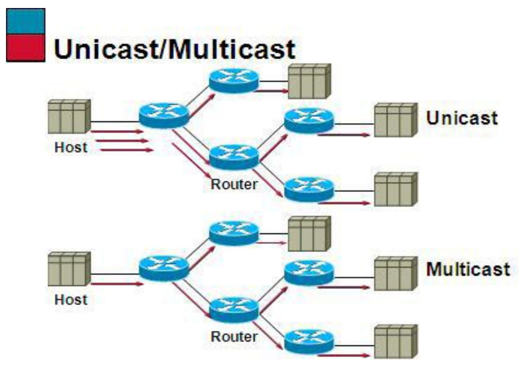

Figure 1.1: Unicast/Multicast Communication through routers

Another important feature of multicasting is its support for data casting applications. Instead of using a set of point-to-point connections between the participating nodes, multicasting can be used for distribution of the multimedia data to the receivers.

II. A SCALABLE GROUP KEY MANAGEMENT PROTOCOL

Our new scalable group key management protocol is based on the following: the Chinese Remainder Theorem and a hierarchical graph in which each node contains a key and a modulus. The protocol is designed to minimize re-key messages, bandwidth usage, encryption, and signature operations. Chinese Remainder Theorem: Let *m*1,*m*2, ...*mn* be *n* positive integers where they are pair wise relatively prime (i.e. *gcd*(*mi*,*mj*)=1 for *i_=j*, 1≤*i*, *j*≤*n*), *R*1,*R*2,...*Rn* be any positive integers, and *M*=*m*1*m*2...*mn*. Then the set of linear congruous equations *X*≡*R*1 *mod m*1, ...*X*≡*Rn mod mn* have a unique solution as: *X*= _*ni*=1 *RiMiyi mod M*, where *Mi*=*M/mi* and *yi*=*M*−1*i mod mi*. In the new protocol, the keys and moduli are constructed as a tree and maintained by the key server. The tree graph is similar to the tree graph in the LKH protocol but each node of the tree in the new protocol is assigned two values: a key and a modulus.

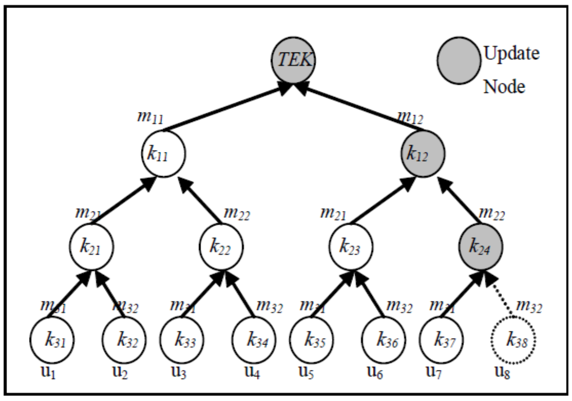

Fig:1

Figure 1 depicts the key and modulus graph, where *TEK* is a traffic encryption key, *kij* is a key encryption key, and *mij* is a modulus. Moduli Maintenance: The key server needs to store 2*log*2*n* moduli and each member needs to store *log*2*n* moduli but they do not need to keep the moduli secret. The sibling nodes in the tree graph are assigned with two different moduli (i.e., *mi*1 and *mi*2 where *i* is the depth of the tree) and the nodes in the different level of the tree are assigned with the different moduli but each a pair of siblings at the same tree depth are assigned with the same two moduli under the different parents (see Figure 1). This means there are only 2*log*2*n* different moduli in the tree graph, i.e. *mij* (1≤*i*≤*log*2*n*, *j*=1, 2) where *i* is the depth of the node in the tree, and the nodes (except the root) on a path from a leaf to the root and its direct children exactly cover all moduli. For instance, in Figure 1, for a path from *u*1 to the root, the moduli on the path include *m*11, *m*21, and *m*31, and the moduli on its direct children include *m*12, *m*22, and *m*32. In addition, all different moduli in the tree graph should be pair wise relatively prime (i.e., *gcd* (*mij, mst*)=1 for *i_=s* or *j_=t*), and each modulus should be bigger than the key encryption value, i.e., *mij*>*Ekil* (*kst*) where *mij* and *kil* belong to the same node and *kst* belongs to its parent node. Key Maintenance: The key server needs to store 2*n*-1 keys,i.e., *TEK* and *kij*(1≤*i*≤*log*2*n*, 1≤*j*≤2*i*) where *i* is the depth of the node in the tree and *j* is the ordinal number of the node in the *i*th depth of the tree, and each member needs to store *log*2*n*+1 keys. The key server shares the keys with each member on the path from its leaf to the root. The keys on its path from the leaf to the root need to be updated in the protocol when a member joins or leaves the group but all moduli must be kept fixed. To update the keys on the tree graph, the key server generates a new key for each update node and encrypts it with its children keys on its path from the leaf to the root. For instance, the key server needs to generate new keys *{TEK_, k_il}* to update *{TEK, kil}* for the arrival of member *ud* (its leaf key is *kwd, w*=*log*2*n*) to the group, where 1≤*i*≤*log*2*n*-1 and *l*=_*d/*2*log*2*n*−*i*_ which is the upper limit integer of *d/*2*log*2*n*−*i*, and encrypts the updated keys using the following formula, where *e*=_*d/*2*log*2*n*−*i*−1_ and *v*=_*d/*2*log*2*n*−1_:

$$
K_{st}=\begin{cases}
E_{k_{st}}(k'_{il}) & \text{if } i{=}log_2 n\text{-}1 \\
 & \text{where } s{=}log_2 n,\ t{=}2l\text{-}1 \text{ or } 2l \\
E_{k_{st}}(k'_{il}) & \text{if } 1{\le}i{<}log_2 n\text{-}1,\ t{\neq}e \\
 & \text{where } s{=}i{+}1,\ t{=}2l \text{ if } e{=}2l\text{-}1 \\
 & \text{otherwise } t{=}2l\text{-}1 \\
E_{k'_{st}}(k'_{il}) & \text{if } 1{\le}i{<}log_2 n\text{-}1,\ t{=}e, \text{ where } s{=}i{+}1 \\
E_{k_{st}}(TEK') & \text{if } t{\neq}v, \text{ where } s{=}1,\ t{=}2 \text{ if } v{=}1 \\
 & \text{otherwise } t{=}1 \\
E_{k'_{st}}(TEK') & \text{if } t{=}v, \text{ where } s{=}1
\end{cases}
$$

The key server then calculates a lock *L* as follows and multicasts the lock with the indices of keys (i.e., *st* in the following formula) to all valid members.

$$L=\sum_{s=1}^{log_2 n}\sum_{t=z}^{z+1} K_{st}M_{sj}y_{sj} \ mod\ M$$

where

$$z=\begin{cases}\lceil d/2^{log_2 n-s}\rceil, & \text{if } \lceil d/2^{log_2 n-s}\rceil \text{ is odd}\\ \lceil d/2^{log_2 n-s}\rceil-1, & \text{otherwise}\end{cases},$$

$$j=\begin{cases}1, & \text{if } t\equiv 1 \ mod\ 2\\ 2, & \text{otherwise}\end{cases},$$

$$M=\prod_{s=1}^{log_2 n}\prod_{j=1}^{2} m_{sj},\ M_{sj}=M/m_{sj},\text{ and } y_{sj}=M_{sj}^{-1}\ mod\ m_{sj}.$$

Each member decrypts the updated traffic encryption key and related key encryption keys based on their own moduli and keys. For the departure of member *ud* from the group, the process is as same as the above except calculating *Kwd* (i.e., *Kwd*=0).

TABLE I : A COMPARISON ON LEVEL OF PROCESSING DIFFICULTY

Protocols	GKMP	Secure Lock	LKH	SGKMP
Level of Processing Difficulty	Low	High	Low	Low

As an illustration, we give the following example for the re-key process in Figure 1, where the member *u*8 requests to join the group. The key server generates new keys *{TEK_*,*k*_12, *k*_24*}* to update *{TEK*, *k*12, *k*24*}* and does the following encryption: *K*38=*Ek*38 (*k*_24), *K*37=*Ek*37 (*k*_24), *K*24=*Ek*_24 (*k*_12),*K*23=*Ek*23 (*k*_12), *K*12=*Ek*_12 (*TEK_*), *K*11=*Ek*11 (*TEK_*). The key server then calculates a lock as *L*=*K*38*M*32*y*32+*K*37*M*31*y*31+*K*24*M*22*y*22+*K*23*M*21*y*21+*K*12*M*12*y*12 + *K*11*M*11*y*11 *mod M*, where *M*=*m*11*m*12*m*21*m*22*m*31*m*32, *Mij*=*M/mij* , *yij*=*M*$^{-1}$*ij mod mij* .

In the protocol, we can see that the key server uses the same modulus (*M*) and parameters (*Mij, yij*) to calculate the lock for any re-key process but the key encryption value (i.e., *Kst*) for calculating the lock are changed based on the re-key requested by the different members. This means the key server can pre-calculate the modulus (*M*) and parameters (*Mij, yij*) to be used for later re-key processing steps and only needs to calculate them once for a fixed tree graph.

III. SCALABILITY OF GROUP KEY MANAGEMENT PROTOCOLS

In order to measure the scalability of group key management protocols more accurately, we propose the following scalability metrics: 'computational complexity', 'bandwidth usage', 'storage', 'number of re-key messages', and 'level of processing difficulty'. Computational complexity measures the processing time in the central key server. Bandwidth usage accounts for the size of total messages sent out by the key server for a re-key process. Storage measures the total size of keys maintained by the key server. The number of rekey messages is the number of such messages needed to be processed by the key server. The level of processing difficulty indicates applicability for small mobile devices. Table I gives a comparison on the level of processing difficulty. Table II gives a comparison of the new protocol with the GKMP, Secure Lock, and LKH protocols without signature protection. Table III gives a comparison with signature protection, where storage and number of re-key messages are the same as in Table II.The signature technique for GKMP and LKH is based upon a single digital signature scheme proposed by Merkle, which has been the most efficient method so far for signing a set of messages destined to different receivers. From Tables II and III, we see that the LKH and SGKMP reduce the encryption operation and bandwidth usage from *O*(*n*) to *O*(*log n*) when compared to the Secure Lock and GKMP protocols, and the length of the lock from *O*(*n*) to *O*(*log n*) when compared to the Secure Lock. In addition, SGKMP has better performance when compared to the LKH protocols. The detailed processing time performance according.

TABLE II: A COMPARISON OF GROUP KEY MANAGEMENT PROTOCOLS WITHOUT SIGNATURE

Scalability Metrics	Computational Complexity		Bandwidth Usage		Storage	Number of Re-key Messages	
Protocols	J	L	J	L		J	L
GKMP	4E	2nE	4N	2nN	(n+2)N	4	2n
Secure Lock	nE+(3n-1)M +(n-1)A +nMR+MD		nN		(2n+1)N	1	
LKH	2log_2nE		2$Nlog_2n$		(2n-1)N	2log_2n	
SGKMP	2log_2nE+ 4log_2nM+ (2log_2n-1)A +MD		2$Nlog_2n$		(2n-1)N+ 2$Nlog_2n$	1	

• J: Join; L: Leave

TABLE III: A COMPARISON OF GROUP KEY MANAGEMENT PROTOCOLS WITH SIGNATURE

Scalability Metrics	Computational Complexity		Bandwidth Usage	
Protocols	Join	Leave	Join	Leave
GKMP	4E+7H +S	2*n*E+S +(4*n*-1)H	6*N* +*L*	$N(2n+log_2 2n)$ +*L*
Secure Lock	*n*E+(3*n*-1)M+(*n*-1)A +*n*MR+MD+H+S		(*n*+1)*N*+*L*	
LKH	$2log_2n$E+($4log_2n$-1)H+S		$2N(log_2n+log_4n)+L$	
SGKMP	$2log_2n$E+$4log_2n$M+MD +($2log_2n$-1)A+H+S		$N(2log_2n+1)+L$	

Please note the following for Tables II and III:

- *N* is the length of the encrypted secret key (default is 128 bits for a symmetric cryptograph) or the length of a hash value (default is 128 bits)
- *L* is the length of the signature
- *n* is the number of members
- H is a hash operation
- E is a symmetric key encryption operation
- S is a signature operation
- A is an Big Integer addition operation
- M is a Big Integer multiplication operation
- MD is a Big Integer modulus operation
- MR is a Big Integer modulus reverse operation to computational complexity is tested in the next section for both with and without the signature operation.

IV. PERFORMANCE OF THE NEW PROTOCOL

The performance of the SGKMP, LKH, and Secure Lock protocols were tested in order to compare their scalability. The testing was done on a PC (Intel Pentium 4 CPU 3.00GHz and 1 GB RAM). The software used for the testing was Java JDK 1.6. The main classes included Java Big Integer, security, and crypto. The encryption algorithm was AES with a 128 bit encryption key, and the signature algorithm was 512 bit RSA. The testing determined the processing time performance according to group size (see Figure 2 and 3) both with and without signature protection. From the testing results, we can see that the Secure Lock has very poor scalability when compared to SGKMP and LKH (Figure 2, where the SGKMP and LKH are in the bottom), and the SGKMP has very good scalability for both with and without signature protection when compared to LKH (Figure 3). The processing time of SGKMP with signature for a rekey request corresponding to a group size of up to a million members is less than 10 milliseconds.

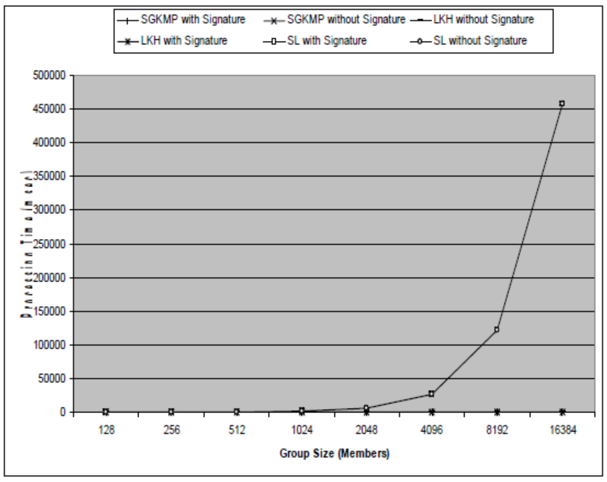

Fig.2 Performance of the SGKMP,LKH, and SL protocols.

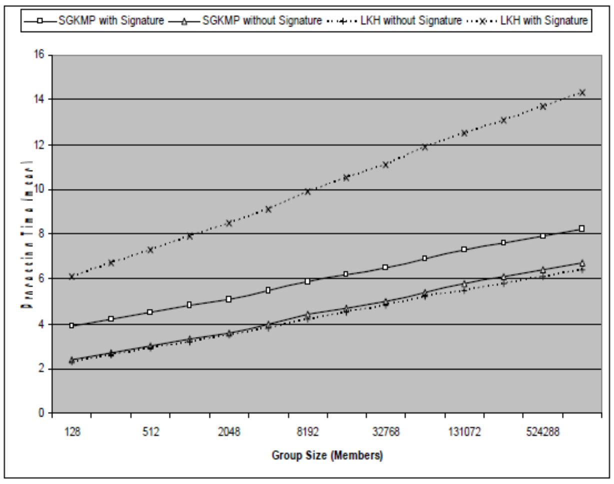

Fig.3. Performance of the SGKMP and LKH protocols.

V. CONCLUSION & FUTURE SCOPE

To improve the scalability of group key management, we propose a scalable group key management protocol and demonstrate that it has better scalability in terms of computational complexity (from testing) and bandwidth usage (from calculations in Tables II and III). The security of PGKMP is mainly based on neighborhood authentication of the nodes, as well as on security associations, while the use of public key cryptography is minimized. PGKMP finds disjoint paths only, so the route discovery cost will be less as compared to LKH where all possible paths exist and a key server has to be maintained. Also due to the double encryption scheme provided to the protocol, the network is more secured. There is a scope to further decrease the overheads and increase more security with this Protocol (PGKMP) and a positive hope for the enhancement of this protocol.

REFERENCES

1. C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Trans. Networking*, vol.8, no.1, pp.16–30, Feb. 2000.
2. D. Wallner, E. Harder, and R. Agee, "Key management for multicast: issues and architecture," National Security Agency, RFC 2627, June 1999.
3. H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) architecture," RFC 2093, July 1997.
4. H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) specification," RFC 2094, July 1997.
5. G. H. Chiou and W. T. Chen, "Secure broadcast using secure lock," *IEEE Trans. Software Engineering*, vol. 15, no. 8, pp. 929–934, Aug. 1989.

Mr. D.CHALAPATHI RAO has received Post Graduation MSc (Maths) from the Department of Mathematics, Andhra University in 2008. He is pursuing Master of Technology in the Dept of Computer Science Engineering, Chaithanya Engineering College, Vishakhapatnam, and JNT University. His research interests are Computer Networking.

Mr. S.VENKATARAO has received Post Graduation MCA from the Department of Computer Science and Engineering, Andhra University in 2008. His research interests are Computer Networking, Data Mining and Data structures