http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (3): 405-408, 2006
ISSN 1812-5638

Performance Analysis of Rekeying Protocols in Multicast Group Key Management

[1]B. Parvatha Varthini and [2]S. Valli
[1]Department of Computer Applications, St. Joseph's College of Engineering, Chennai-119, India
[2]Department of Computer Science, College of Engineering Guindy, Anna University, Chennai-25, India

Abstract: Group communication is the basis for many recent multimedia and web technology applications. Compared to unicast, Multicast faces a great challenge of providing security requirements like data confidentiality, data integrity and source/group authentication. The primary security mechanism of group communication is achieved by encrypting and decrypting the data with a common shared cryptographic key known as the group key. The group key management is one of the most critical problems in a large dynamic group. There are different key management algorithms that facilitate efficient distribution and rekeying of the group key. All rekeying protocols add communication overhead as well as computation overhead at the Group Key Controller (GKC) and at the group members. This study discusses the performance of various key management algorithms with respect to computational and communication overheads that incur in the group members.

Key words: Group Key Controller (GKC), cryptographic group key, logical key graph, one way function tree

INTRODUCTION

Data integrity, data confidentiality and group authentication are the minimal multicast security requirements. These requirements are achieved by encrypting multicast messages using a cryptographic Traffic Encryption Key (TEK) known as the group key. The GKC (Group Key Controller) generates and distributes the group key to all group members. The group key must be changed from time to time as well as for every membership change to safeguard its secrecy.

The GKC changes and distributes the group key using a specific group key management algorithm. Any group rekey algorithm must provide Backward Secrecy which guarantees that a passive adversary who knows a subset of old group keys cannot discover subsequent keys and Forward Secrecy which guarantees that a passive adversary who knows a subset of group keys cannot discover preceding group keys. These algorithms introduce communication and computational overhead at the GKC as well as at every group member while computing the new group key during join/leave operation.

In this research a detailed study of existing group rekeying algorithms including the one proposed by the authors of this work is carried out to find an efficient group rekeying algorithm resulting in minimal overhead. The existing group rekeying protocols are classified as Simple Rekeying Approach, Hierarchical key Graph algorithm and Periodic Rekeying algorithm. This study estimates initial group key gereration overhead, join-rekey overhead and leave-rekey overhead of each group rekeying algorithm. This research compares the performance of these group rekeying algorithms with respect to communication and computational overheads incurred by them.

SIMPLE REKEYING PROTOCOLS

n nodes-one controller: In the simple group rekeying scheme (Wallner *et al*., 1999), all members are connected to a GKC. The GKC generates and encrypts the group key separately for every member of the group. When a member joins/leaves the group, the GKC creates a new key. The GKC encrypts and sends this new key to every member separately. Here both computational and communication complexity is linearly proportional to the group size. This algorithm is inefficient when the group size is too large.

Group Key Management Protocol (GKMP): In GKMP (Harney and Muekenhirn, 1997), initially, the GKC selects a member and initiates the creation of a Group Key Packet (GKP). The packet contains the current Group Traffic Encryption Key (GTEK) and a key (GKEK) to deliver the future GTEK. To handle future rekeys, the GKC then creates a digitally signed Group Rekey Packet (GRP), which consists of the earlier created GKP encrypted with the GKEK. When a member joins, the GKC selects a

Corresponding Author: B. Parvatha Varthini, Department of Computer Applications, St. Joseph's College of Engineering, Chennai 119, India 600119 Tel: 91-44-22273146 Fax: 91-44-24500861

member and creates a new GKP containing a new GTEK. In addition, it creates a new GRP, which is encrypted under the earlier next GKEK. This method fails to maintain the forward secrecy when a member leaves since every member knows the GKEK.

HIERARCHICAL KEY GRAPH ALGORITHMS

Here the group is viewed as a logical tree structure to reduce the overhead incurred at the group members during the join/leave operation. The GKC creates a rooted balanced tree that has as many leaf nodes as there are members. Each leaf node of the key tree is associated with a member of the group. Each internal node represents a logical subgroup. The root node represents the group key.

Logical Key Hierarchy Graph (LKH): LKH (Wallner *et al*., 1999) is an efficient rekeying method for large groups. It uses a hierarchical key tree. The GKC shares a separate unique secret key with each one of the members through unicast. GKC creates all internal node keys but sends only a subset of them to each member. Each member receives the keys of all nodes in the path from that member to the root. The number of keys that each member holds is equal to log n, where n is the size of the group. When a member joins the group, the GKC needs to change every internal node key in the new member's path to the root. Hence the GKC changes log n keys. The GKC encrypts and sends each of these internal node keys twice: once for the new member and once for the existing member. Hence the GKC does 2 log n encryptions and transmissions. When a member leaves the group, the GKC changes only the keys that the leaving member knows. Hence the GKC rekeys log n keys and encrypts each key twice: once per each child of the internal node. Leave rekeying requires 2 log n encryptions.

One way Functional Tree (OFT): An improved key management scheme (Waldovogel *et al*., 1999) known as OFTis used for key establishments using OFT in large dynamic groups (Alan *et al*., 2003; Ku and Chen, 2003). It also uses a logical key tree used for rekeying with better leave rekeying compared to LKH. The GKC shares a separate unique secret key with each one of the members through unicast. The GKC uses a one-way function 'g' to compute a blinded key corresponding to each key in the key tree. Each member receives the blinded keys of the sibling of the internal nodes in the path from that member to the root of the tree. Each internal node key is computed by applying a mixing function 'f' to the blinded keys of its children. When a member joins, the GKC sends to the new member log n blinded keys. Further, the GKC sends log n+1 blinded keys to the other members. Thus, the GKC encrypts and sends 2 log n +1 blinded keys, when a member joins the group. When a member leaves, the GKC needs to send as many blinded keys as the length of the path from the rekeyed node to the root. Thus, it sends log n new blinded keys to the group members and the rekeyed node. Hence, the GKC sends 2log n +1 keys including one unblinded key to the rekeyed node. The GKC performs $\log_2$ n one-way function computations to find the blinded keys of these new keys.

Efficient Large-Group Key (ELK): The ELK protocol (Perrig *et al*., 2001) uses a hierarchical tree and is very similar to the OFT in the sense that a parent node key is generated from its children keys. ELK uses Pseudo-random Functions (PRFs) to build and manipulate the keys in the hierarchical tree. A PRF uses a key K on input M of length m to generate output of length n represented by the following notation: $PRF^{m->n}(M)$. ELK does not require any multicast message during a join operation. When members are deleted, new keys have to be generated for those nodes in the path from the removed node to the root.

Enhanced OFT: Our approach EOFT (Parvathavarthini and Valli, 2005) enhances the efficiency of OFT by using the Chinese reminder theorem. It follows hierarchical key graph for rekeying. This algorithm uses CRT tuple of smaller remainders instead of keys of larger bits. Any computation on keys is carried out in parallel on their corresponding CRT remainders. This algorithm reduces the computational overhead of a leave/join event compared to OFT. The GKC shares a pseudo function P, a random number M, an Arithmetic function A along with a separate unique secret key with each member through unicast. A pseudo function P which is a one-way hash function, is used to generate pseudo keys corresponding to each key. P can even be a mac function or simple transformation of bits depending on the security level required. The GKC generates a random number M = [mi] for finding mod mi reminder tuples corresponding to each key. Each member receives the CRT tuples of pseudo keys of the sibling of the internal nodes in the path from that member to the root of the tree. Each internal node's key is computed by applying an arithmetic function R on its children's pseudo key tuples. The arithmetic function R can be +/-/*/ a repeated combination of odd number of these three operations. But existing works use only hash for P and XOR for R.

When a member joins the group, the GKC creates a new node in the tree so that the tree is balanced. This is achieved by splitting the nearest leaf node from the root. After accommodating the new member in the tree, the GKC shares a unique secret key, new R and M with the new member through unicast. The random number M is derived from the previous number by omitting any one of its factors. The new arithmetic operation R is a new combination of odd number of operators +, * and -. The GKC just informs the new R & M through a multicast message encrypted with the old group key to the existing group members. Unlike in OFT and LKH, the GKC does not send any new internal node keys to the existing members in the join event. So the join event communication overhead is highly reduced in EOFT. When a member leaves the group, the GKC reconstructs the tree as a balanced tree. If the departing member's sibling is a leaf node, it gets associated with its parent node or it assumes the parent's position on the tree. The GKC needs to rekey to maintain forward secrecy. Unlike in LKH, the GKC does not change all the pseudo keys that the departing member knows. Since the GKC does not use pseudo keys to encrypt any message, it does not change all the pseudo keys supplied to the departing member. For the sibling leaf node of the departing member, the GKC sends a new key encrypted with its old key. The GKC rekeys all the keys from rekeyed node's position to the root.

BATCH AND PERIODIC REKEYING

In a large and highly dynamic group, immediate rekeying policy may result in frequent rekeyings and might overwhelm the GKC computationally. To reduce the rekeying overhead, the GKC may choose to rekey a group periodically (Briscoe, 1999) or process group membership changes in batches (Li *et al*., 2001). Commercial applications such as Internet or satellite-TV distribution use batch or periodic rekeying. In batch rekeying the departing members continue to get access to the group data for a brief period after they leave so that forward secrecy fails. The joining members are put on hold until the next rekeying instance. In MARKS (Briscoe, 1999), the group members may join the group at any time, but A member joins the group and assures its life time as an interval between t_1 and t_2. GKC supplies the group keys K_{t1}, K_{t1+1}, … K_{t2-1},K_{t2}. GKC starts with a random seed at the root and computes child's seed using two different functions g_L and g_R, recursively until leaf nodes. During registration, the GKC sends seeds that are necessary for a member of the group to compute group keys for its entire life duration in the group. The advantage of this protocol is that all the keys are transported through unicast during registration. The disadvantage is rekeying complexity at the time of leaving.

CENTRALIZED FLAT TABLE

It changes the hierarchical tree to a flat table (Chang *et al*., 1999) with the effect of decreasing the number of keys held by the GKC. The table has one entry for the TEK and 2w more entries for KEKs, where w is the number of bits in the member id. There are two keys available for each bit in the member id, one associated with each possible value of the bit. A member knows only the key associated with the state of its bit. In total, each member holds w + 1 keys. For example, a member with id 0101 knows $KEK_{0.0}$, $KEK_{1.1}$, $KEK_{2.0}$ and $KEK_{3.1}$. When a member leaves the group, all the keys known by it are changed and the KDC sends out a message that contains two parts. The first part has the new TEK encrypted with each unchanged KEK. In the second part, each of the new KEKs is encrypted with its old KEK and with the new TEK. In this way, every remaining member can update its old KEKs without gaining further knowledge about the KEKs of the other members. This scheme is susceptible to collusion attacks. A set of evicted members, which have IDs with complementary bits, may combine their sets of keys to recover a valid set of keys and thereby gain unauthorized access to group communication.

RESULTS

The comparison of performance of the proposed EOFT with the other group key management schemes is done quantitatively and tabulated below. Table 1

Table 1: Computational cost comparison with respect to processing needed for join and leave operation

	At the GKC		At member join					
Algorithm	Join	Leave	Others				New member	Leave others
n node↔1 GKC	nE	nE	D	D	…	D	D	D
GKMP	2E	nE	D	D	…	D	D	D
LKH	(2d+1)E+(d+1)G	2dE+dG	dD				(d+1)D	2dD
OFT	2G+(d+2)g+(2d+1)E+df	df+G+dg+(d+1)E	D+g+dD+df				D+g+df	dD+g+df
EOFT	dR+2E	G+dP+dR+(d+1)E	D+dR				D+P+dR	dD+P+dR

n = Size of the group, I = Number of bits in the member id, T = The count of time during when the group exists, G = Key generating function,
t = The count of time during when the member persists, g = Infeasible function, d = Height of the tree = log n if tree is balanced, f = Mixing function
E = Encryption operation, P = Pseudo function, D = Decryption operation, R = +/-/*/ a repeated combination of odd number of *,+,-

Table 2: Communication cost

Group key management algorithm	Join messages from the GKC to Other	Join messages from the GKC to New member	Leave from messages the GkC
Simple			
n nodes↔1 GKC	(n-1)	1	N
GKMP	2	2	New
Hierarchical key graph			
LKH	d	d+1	2d
OFT	d+1	d+1	D+1
ELK	0	d+1	D+1
EOFT	1	d+1	D+1
Periodic rekeying			
Marks	-	t	2 log(t/2)
Centralised table			
FT	2I	I+1	2I

Table 3: Security aspects

Algorithm	Backward	Forward
Simple		
n node↔1 GKC	√	√
GKMP	√	X
Hierarchical key graph		
LKH	√	√
OFT	√	√
ELK	√	√
EOFT	√	√
Periodic rekeying		
Marks	√	√
Centralised table		
FT	√	√

compares the computational complexity of the group key management schemes. The computational complexity depends on the number of operations needed for deriving, generating and forwarding the group key. This complexity is incurred in the group members as well as in the GKC. It includes the parameters such as size of the group, height of the tree, number of cryptographic operations and the evaluation of generating functions. It shows that leave rekeying computational complexity of EOFT is half of LKH. The measure of communication complexity depends on the number and size of the messages transmitted for key updates. Table 2 shows the number of key update messages sent during join operation of a group member. EOFT reduces join rekeying communication complexity to simply one multicast message compared to all other group rekeying algorithms. The leave rekeying communication complexity is EOFT in just half that of LKH. Table 3 presents the security properties provided by the group key management algorithms.

DISCUSSION

Simple rekeying algorithm lacks scalability. GKMP fails to meet Forward serecy. Centralized flat table suffers from collision attack, which reveals some bits of the key. The periodic rekeying algorithm needs the departure time of group members during the time of joining itself, which is not possible to predict always. Hierarchical key graph algorithms solve scalability issue. ELK suffers from collusion attack. EOFT approach uses smaller size CRT key tuples instead of larger bit keys so that computations are done in parallel and short rekey messages are framed. The authors' contribution EOFT overcomes the shortcomings of the existing rekeying algorithms. EOFT provides flexibility in selecting the group key generation functions. Multicast applications can use the proposed EOFT algorithm because they can reduce the cryptographic latencies if the required level of security is minimal. EOFT provides a better performance compared to other key management algorithms without sacrificing security. In future, efficient group communication architecture that overcomes scalability issue and cryptographic latencies is to be framed independent of group key management algorithms.

REFERENCES

Alan, T.S. and D.A. Mcgrew, 2003. Key establishment in large dynamic groups using one-way function trees. IEEE Transaction on Software Engineering, 29: 444-458.

Briscoe, B., 1999. MARKS, Zero Side Effect Multicast key management using arbitrarily revealed key sequences. In Proceedings of International Workshop on Networked Group Communications, Pisa, Italy, pp: 301-320.

Chang, I., R. Engel, D. Kandlur, D. Pendarakis and D. Saha 1999. Key management for secure Internet multicast using Boolean function minimization technique. In IEEE Infocom., 2: 689-698.

Harney, H. and C. Muckenhirn, 1997. Group Key Management Protocol (GKMP) Specifcation. RFC 2093, 2094.

Ku, W.C. and S.M. Chen, 2003. An Improved Key Management Scheme for Large Dynamic Groups Using One-Way Function Trees. Proc. Oct., ICPPW'03, pp: 391-396.

Li, X.S., Y.R. Yang, M.G. Gouda and S.S. Lam, 2001. Batch rekeying for secure group communications. In Proceedings of 10th International Conference World Wide Web, pp: 525-534.

Parvatha, V.B. and S. Valli, 2005. EOFT: An enhanced one way function tree rekey protocol based on chinese remainder theorem. In Proceedings of 20th International Symposium on Computer and Information Science ISCIS 05 Lecture Notes on Computer Science, 3733: 33-44.

Perrig, A., D. Song and J.D. Tygar, 2001. ELK: A new protocol for efficient large-group key distribution. In Proceedings of the IEEE Symposium on Security and Privacy. IEEE Computer Society Press, Los Alamitos, Calif, pp: 247-262.

Waldvogel, M., G. Caronni, D. Sun, N. Weiler and B. Plattner, 1999. The Versa Key framework: Versatile group key management. IEEE J. Selected Areas in Commnni., pp: 1614-1631.

Wallner, D., E. Harder and R. Agee, 1999. Key Management for Multicast: Issues and Architectures. RFC 2627.