ISSN:0975-9646

V.Jaya Ramakrishna et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (5) , 2012,5265-5271

Secure Analysis for Interval-based Algorithms

V.Jaya Ramakrishna,K.Nithin Babu,M.N.Satish Kumar

Department of Computer Science & Engineering
Gudlavalleru Engineering College , Gudlavalleru

Abstract-we consider several distributed collaborative key agreement and authentication protocols for dynamic peer groups. There are several important characteristics which make this problem different from traditional secure group communication.
They are: 1) distributed nature in which there is no centralized key server; 2) collaborative nature in which the group key is contributory (i.e., each group member will collaboratively contribute its part to the global group key); and 3) dynamic nature in which existing members may leave the group while new members may join. Instead of performing individual rekeying operations, i.e., recomputing the group key after every join or leave request, we are going to re key for a batch of join and leave operations.
The objectives of the project are to generate a group key.With the help of the group key sharing the resources like accessing the files and implement the Queue-batch algorithm, which performs the best among the three interval-based algorithms by comparing them. More importantly, to show that Queue-batch algorithm can substantially reduce the computation and communication workload in a highly dynamic environment

1. INTRODUCTION

The mainstay of the project is to collaboratively generate a common key for peer to peer group communication. To dynamically perform re-keying operation after batch of joins or leaves using Queue Batch algorithm and to share resources using the generated group key.
The purpose of the proposed system is to provide the members of a group with secure common group key. This group key is generated collaboratively wherein each node becomes a part of the key generation.
The distributive nature of the proposed system, avoids the usage of a centralized key server. The dynamic nature of the system allows the existing members to leave the group while new members can join, instead of performing individual rekeying operations.
The system uses Queue-batch algorithm for re-keying. The algorithm can substantially reduce the computation and communication workload in a highly dynamic environment. The group key is used for future communication among the members of the group.
Other than Queue-batch algorithm we have Re-build algorithm and Batch algorithm .but the last two algorithms are not as effective as Queue –batch algorithm because Queue-batch works more efficient than the other algorithms at re-keying when no element leaves from the group. ,i.e, the element which is entered newly is kept in a Queue sub-tree phase and next the element is added to the group when an element leaves through Queue.
In general the problems with the existing system are Key information depends on centralized key server and Computational and Communication cost is more.And when coming to re-keying , Individual re-keying is done Whenever a member joins or leaves in the case of distributed key generation algorithm. More resources used for re-keying because it is done for each join or leave operations.
So to avoid these problems we use the Queue-batch algorithm for re-keying. The algorithm can substantially reduce the computation and communication workload in a highly dynamic environment. The group key is used for future communication among the members of the group.

2. SYSTEM STUDY

We consider several distributed collaborative key agreement and authentication protocols for dynamic peer groups. There are several important characteristics which make this problem different from traditional secure group communication. They are: 1) distributed nature in which there is no centralized key server; 2) collaborative nature in which the group key is contributory (i.e., each group member will collaboratively contribute its part to the global group key); and 3) dynamic nature in which existing members may leave the group while new members may join. Instead of performing individual rekeying operations, i.e., recomputing the group key after every join or leave request, we discuss an interval-based approach of rekeying. We consider three interval-based distributed rekeying algorithms, or interval-based algorithms for short, for updating the group key: 1) the Rebuild algorithm; 2) the Batch algorithm; and 3) the Queue-batch algorithm. Performance of these three interval-based algorithms under different settings, such as different join and leave probabilities, is analyzed. We show that the interval-based algorithms significantly outperform the individual rekeying approach and that the Queue-batch algorithm performs the best among the three interval-based algorithms. More importantly, the Queue-batch algorithm can substantially reduce the computation and communication workload in a highly dynamic environment. We further enhance the interval-based algorithms in two aspects: authentication and implementation. Authentication focuses on the security improvement, while implementation realizes the interval-based algorithms in real network settings. Our work provides a fundamental understanding about establishing a group key via a distributed and collaborative approach for a dynamic peer group.

3. CONVENTIONAL SYSTEM

The existing system involves either centralized key server (in which all the systems depend on centralized server for key generation), and individual rekeying is done for join or leave operations in case of distributive key generation algorithms. In case of individual re-keying, after every join or leave operation each member individually rekey's. More resources are used for re-keying because it is done for each

join or leave operations. In case of using a centralized server, the risk of single point failure is more.

DRAWBACKS OF CONVENTIONAL SYSTEM

- Key information depends on centralized key server.
- Computational and Communication cost is more.
- Individual re-keying is done. Whenever a member joins or leaves in the case of distributed key generation algorithm.
- More resources used for re-keying because it is done for each join or leave operations.

4. PROPOSED SYSTEM

The proposed system involves collaborative key agreement in which all nodes become a part of the secure group key. Moreover, rekeying is done after a batch of join or leave operations. The protocol remains efficient even when the occurrences of join/leave events are very frequent. Here Key information does not depend on centralized key server. So it is free from the problem of single point failure. Computational and Communication cost is less. Resources used for rekeying is minimized because it is being done for batch of join/leave operations.

Group key agreement schemes:

Based on the Diffie-Hellman protocol [2], where all arithmetics are performed in a group of prime order p with generator a: the blinded key of node v can be generated by

$$BK = aK''\bmod p$$

The group key is generated in a shared and contributory fashion and there is no single point of failure The contributions of our work are:

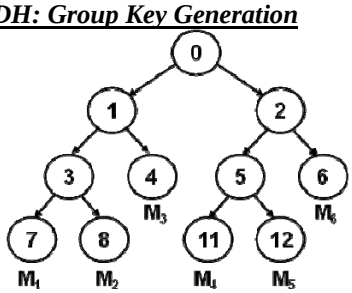
The key agreement protocol is distributed in nature and does not require a centralized key server.

The key agreement protocol is contributive – each member contributes its part to the overall group key.

We illustrate that instead of performing individual rekeying operations, one can use an interval-based approach to significantly reduce the computation and communication costs of maintaining the group key.

We propose three distributed interval-based rekey protocols. and carry out qualitative and simulation-based analysis to illustrate their performance merits.

TGDH: Group Key Generation

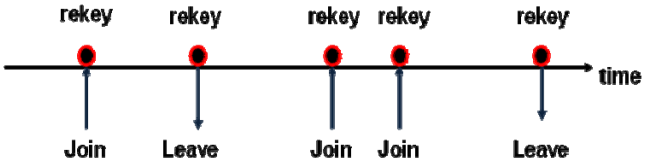


E.g., M_1 generates the group key via:

- K_7, BK_8 → K_3
- K_3, BK_4 → K_1
- K_1, BK_2 → K_0 (Group Key)

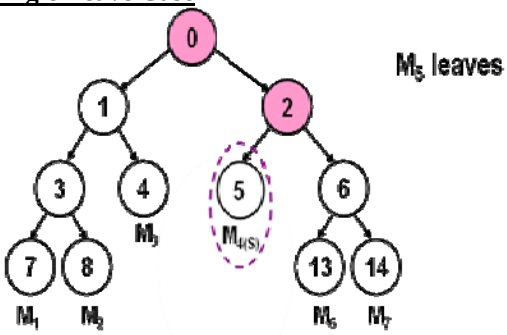
TGDH: Membership Events

Rekeying (*renewing the keys of the nodes)* is performed at every single join/leave event to ensure backward and forward confidentiality.



Cases for nodes Leaving and joining dynamically

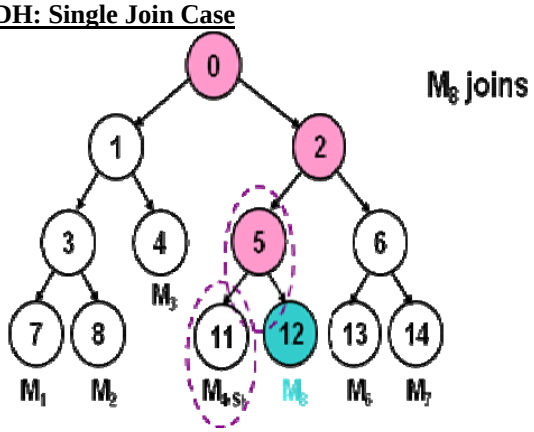
TGDH: Single Leave Case



M_4 becomes the sponsor. It rekeys the secret keys K_2 and K_0 and broadcasts the blinded key BK_2.

- M_1, M_2 and M_3 compute K_0 given BK_2.
- M_6 and M_7 compute K_2 and then K_0 given BK_5.

TGDH: Single Join Case



- M_8 broadcasts its individual blinded key BK_{12} on joining.
- M_4 becomes the sponsor again. It rekeys K_5, K_2 and K_0 and broadcasts the blinded keys BK_5 and BK_2.
- Now everyone can compute the new group key.

Description of Algorithms

In this subsection, we present three interval-based distributed rekeying algorithms. They are the *Rebuild algorithm*,the *Batch algorithm* and the *Queue-batch algorithm.* The use of interval-based rekeying aims to maintain good rekeying performance, independent of the dynamics of joins and leaves. The three distributed algorithms are developed based on the following assumptions:

The key tree of TGDH is used as a foundation of all the algorithms.

The rekeying operations are carried out at the beginning of every rekey interval. There exists a *virtual queue* holding all join and leave requests till the beginning of the next rekey interval. When a new member sends a join request, it should also include its individual blinded key. For simplicity, all clients know the existing key tree structure and they also know all the blinded keys within the tree structure. The group members would elect sponsors to be responsible for computing and broadcasting blinded keys. To obtain the blinded keys of the renewed nodes (a node is said to be ***renewed*** if it is a non-leaf node and its associated keys are updated), the key paths of the sponsors should contain those renewed nodes. Since the interval-based rekey ing operations involve nodes lying on more than one key paths, more than one sponsors may be elected. Also, a renewed node may be rekeyed by more than one sponsor. In this case, we assume that the sponsors can coordinate with one another such that the blinded keys of all the renewed nodes are only broadcast once.

We adopt the following notations for the three distributed algorithms. Let T denote the existing key tree. Assume that L *2* 0 existing members ***M' = (Ad:,*** . . . , *ML)* wish to leave, and J *2* 0 new members ***~j = (M;***, . . . , *M:)* wish to join the communication group within a rekey interval.

Rebuild Algorithm

The motivation for the rebuild algorithm is to *minimize* the final tree height so that the rekeying operations for each group member can be reduced. At the beginning of every rekey interval: we reconstruct the whole key tree with all existing members who remain in the group, together with the newly joining members. The resulting tree would be a ***cornplete*** tree. The pseudo-code of the Rebuild algorithm to be performed by every member is shown below:

Rebuild (T. M ~ ,J , M', L)
1. obtain all members from T and store them in M ' ;
2. remove the L leaving members in M' from M ' ;
3. add the J new members in MJ to M ' ;
4. create a new binary tree T' based on members in M' and set
T = T';
5. rekey the key nodes and broadcast the new blinded keys in T ;

Figure illustrates the scenario that members ***Ad2,*** *M5* and *M7* wish to leave the communication group and a new member *Ma* wishes to join the group. The resulting key tree has five members and all the nodes need to be renewed. The sponsors will include all the five members.

M,,M,,M, leave
M l ~ M~ 3l 1 ~ j
M, M,

Batch Algorithm

The Batch algorithm is based on the centralized approach in *[6],* except that we are now applying it to a distributed system without a centralized key server and all clients contribute to the composition of the group key. The pseudo code of the Batch algorithm is given as:

Batch (T, M 3 . J , M', L)
I . i f (L = = O) { / * **pure join case** * /
2. create a new tree T' based on new members in M j ;
3. either (a) add T' to the shallowest node of T (which need not be the leaf node) such that the merge would not increase
the height of the result tree, or (b) add T' to the root node of T if the merge to any node of T would increase
the tree height:
4.)else { / * L > 0 * /
5. sort M' in an ascending order of the associated node IDS of the members and store the results in M'." = (Mi", . . . , M i S) ;
6. i f (L *2* J) {
7. /* **more members want to leave than join** * /
8. if (J > 0)
9. replace the departed nodes of (Mi2', . . - , *M$')* with J loined nodes;lo. i f (L - J > O)
{
1 I. remove remaining L - *J* leaving leaf nodes to the parent node:
12. promote the siblings of the leaving leaf nodes;
13 1
14) else {
15. /* **more newly joining members than leaving members** */
16. divide MJ into L subgroups G = (GI, . . . , GL) such that the fi rst J mod L subgroups (G, . - . , GJ mod L) contain + 1 new members and the rest contain new members;
17. create L subtrees (T;, . . . ,TL) for the subgroups G ;
18. replace the departed nodes of (Mi3', . . . , M:, .) with the roots of (T l , . . . ,T; ,,, .) and the remaining departed nodes with the roots of remaining subtrees;
19. elect the members to be sponsors if (1) they are new members, or (2) the rightmost members of the subtrees rooted at the siblings of the departed nodes or replaced nodes in T ;
20. if (sponsor)
21. rekey the key nodes and broadcast the new blinded keys; Notice that the sponsors may have to wait for the blinded keys on another key path in order to proceed upwards to rekey the nodes. Finally, all the members obtain the necessary blinded keys to compute the new group key ***KO.***

The Batch algorithm is illustrated with two examples. In Figure *5,* we illustrate the case L > J > 0 of the Batch algorithm. Suppose *M2*, *M5* and *M7* leave and a new member *Ma* wishes to join. The following steps will be carried out:

(i) *Ma* broadcasts its join request, including its individual blinded key.
(ii) (ii) The leaf node 6 associated with *M7* is replaced by the node of *Ma,* and the leaf nodes 8 and 22 are removed. Nodes 7 and 23 are promoted to nodes **3** and I I , respectively. (iii) *M1, M4, M6* and *M8* are selected to be the sponsors. *M1* rekeys secret keys *K1* and *KO* and *Mq* rekeys *K5, K2* and *KO. Adl* then broadcasts *BIC1* and *M4* broadcasts *BIG* and *BK2. M6* and though having the sponsor role, do not need to broadcast any blinded keys as *M4* has already broadcast this information. (iv) Finally, every member can compute the group key based on the received blinded keys.

The Batch algorithm whereL> J>O

The case **J** > **L** > 0 of the Batch algorithm. Suppose *Ma, Ad9* and *MIo* join, and *M2* and *M7* leave. The rekeying process is: (i) *Ada, Mg,* and *Adlo* broadcast their join requests together with their own individual
blinded key. (ii) *Ad8* and *Mg* form the subtree *Ti* and *Mlo* is the only member of the subtree *Ti.* The root of
Ti replaces node *6* and the root of *Ti* replaces node 8. (iii) The sponsors will be *MI, Ad6, Adg* and *Adlo.* (iv) *M8* and *Ad9* first need to compute the secret key *Kg,* and either one of them can compute and broadcast the new blinded key

BK6. (v) *MI* (or *Mlo)* rekeys *I(3* and *K1* and broadcasts *BIG* and *B K l* . n/ls rekeys *1C2* and broadcasts *BK2.* (vi)
Finally, all the members can compute the group key *KO.*

The Batch algorithm

where J > L > 0

Queue-batch Algorithm

The previous approaches perform rekeying at the beginning of every rekey interval, which can result in a high processing load during the update instance and thereby delay the start of the secure group communication. The processing load includes the computation cost of the exponentiation operations in generating the keys, as well as the communication cost of broadcasting all the blinded keys to all members in the communication group. We propose a more effective algorithm which we call the *Queue-batch* algorithm. The intuition of this algorithm is to reduce the rekeying load by pre-processing the joining members in the virtual queue during the idle rekey interval.
The Queue-batch algorithm is divided into two phases, namely the *Queue-subtree formation* phase and the *Queuemerge*
phase. The first phase occurs whenever a new member joins the communication group during the rekey interval.
In this case, we append this new member in a temporary key tree *T'.* The second phase occurs at the beginning of every rekey interval and we merge the temporary tree *T'* (which contains all newly joining members) to the existing key tree *T.* Specifically:

Queue-subtree (T')
1. if (a new member joins) {
2. if (T' == NULL) / * no new members in T' * /
3. create a new tree T' with the only one new member;
4. **else** { / * there are new members in T' */
5. fi nd the insertion node;
6. add the new member to T';
7. elect the rightmost member under the subtree rooted at the sibling of the joining node to be the sponsor;
8. if (sponsor)
9. rekey the key nodes and broadcast the new blinded keys to the communication group;
lo. }
11. }

Queue-merge (T, T', **M',** L)
I . i f (L = = O) (/ * there are no leave * /
2. add T' to either (a) the shallowest node (which need not be the leaf node) of T such that the merge would not increase the resulting tree height, or (b) the root node
of T if the merge to any locations would increase the resulting tree height;
3. } **else** / * there are leaves * /
4. add T' to the highest leave position of the key tree T ;
5. elect members to be sponsors if they are (a) the rightmost member of the subtree rooted at the sibling nodes of the departed
leaf nodes in T , or (b) the rightmost member of T';
6. **if** (sponsor)
7. rekey the key nodes and broadcast the new blinded keys to the communication group;

The Queue-batch algorithm is illustrated in where members *Ma, Mg* and *Mlo* wish to join the communication group, while *M2* and *M7* wish to leave. Then the rekeying process is as follows: (i) At the *Queue-subtree formation* phase, the three new members *&I8, Mg,* and *Mlo* would first form a tree *T'. Mlo,* in this case, will be elected
as the sponsor. (ii) At the *Queue-merge* phase, the tree *T'*

The Queue-merge phase

will be added at the highest departed position, which is at node 6. Also, the blinded key of the root node of T', which is BKs, is broadcast by Mlo. (iii) The sponsors, Mi, ***Ad6,*** and Mlo, are elected. (iv) Ml rekeys the secret key ***IC1*** and broadcasts the blinded key BK1: M6 rekeys the secret key K2 and broadcasts the blinded key BK2. (v) Finally, all members can compute the group key.

Performance Evaluation

In this section, we present the mathematical analysis of the three proposed algorithms. We consider two performance measures, namely:
I . *A~~eragileui nber of r e t ~we dn odes:* a node is said to be *reriewed* if it is a non-leaf node and its associated keys are renewed. This metric provides a measure of the communicatioii cost since new blinded keys of the renewed nodes have to be broadcast to the whole group.
2. Average rzurnber of e~~orzetitiatioonp erations: this metric provides a nieasure of the coinputation load for all members in the communication group.
For simplicity, we assume the following in the analysis:
The existing key tree T is a completely balanced tree before the interval-based rekeying event. Each member has a homogeneous leave probability.
The number of blinded key computations simply equals that of renewed nodes, provided that the blinded key of each renewed node is broadcast only once. For the mathematical analysis, let N be the number of members originally in the system, L (where L *5* N) be the number of members which wish to leave the system, and J > 0 be the number of new members which wish to join the communication group. Let T denote the existing tree which contains N members. The level of a node v is 1 = Llog2(v+ I)], where v is the node ID, and the maximum level of T is h. Based on the first assumption, we know that N = 2h. Also, let Ralg be the number of renewed nodes and &,lg be the number of exponentiations for the particular algorithm *alg.* The performance measure Ea19 is composed of two parts: &ilagnd &,big, which represent the number of exponentiations of calculating the secret keys (which is
done by all members) and the number of exponentiations of calculating the blinded keys (which is done by sponsors
only). We have Based on the last assumption. we know the number of blinded key computations is In the following analyses, we only consider the number of secret key computations &,Sig .

Analysis of the Rebuild Algorithm

Given N, L and J, we can obtain the *exact* expressions for the two performance measures RReblLainldd even if the existing key tree T is not completely balanced originally.
The resulting number of members is N* = AT- L + J > 0. Thus, the number of renewed nodes (i.e. the number of non-leaf nodes) is For EReblLiid(Nf). we find that when N* *5* 1, ERebuild(Nf) = 0. If Nf E (2h'-1, 2h'] for hr > 1 where
h' = Llog2(Nf - 1) J + 1, we have &AebTLild(Nf=) (number of members at level h') x h' + (number of members

at level h' - 1) x (h' - 1) - 2 (N* **-2L'0gz(N*-1)J)(~log2(~-*** 1)J + 1) + (N*- ~ (N * - ~ L ' o ~ z (~ ' - ~L)loJg)2)(N* - 1) J = NfLlog2(N* - 1)J +2~*-2(L'ogz(N*-l)J+l).

Analysis of the Batch Algorithm

In analyzing the performance of the Batch algorithm, we consider the following five cases. Note that when L > 0, the performance metrics will depend on the membership leave positions and exact metrics cannot be obtained. Therefore, whenever L > 0, we derive the *expected* performance measures. We also define RaLga,,n d &,lg,, be the two performance measures under condition c. We also adopt the convention that the combination (:) equals *0* if n < 0, r < 0 or n < *r*. Due to limited space, readers can refer to *[5]* for detailed mathematical derivation and results.

Analysis of Queue-batch Algorithm

The main idea of the Queue-batch algorithm exploits the idle rekey interval to pre-process certain rekeying operations.

When we compare its performance with the Rebuild or Batch algorithms, we only need to consider the rekey operations occurring at the beginning of each rekey interval. When J = 0, Queue-batch is equivalent to Batch in the pure leave scenario. For J > 0: the number of renewed nodes in Queue-batch during the Queue-merge phase is equivalent to that of Batch when J = 1. Thus, the expected number of renewed nodes is - L, if J = 0 and L > 0 if J > 0 and L > 0. Also, the expected number of exponentiations when J > 0 for Queue-batch is given by E [E~a t ch,LJ>= o], i f J = O a n d L > O

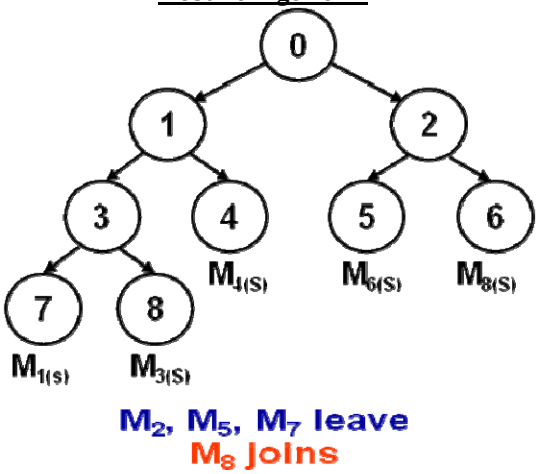
EIEBatch.J=l and L>O] - + dJ: if J > Oand L > 0. (8)

For J > 0 and L > 0, assume the new subtree is attached to a node at some level d. We first decrement d from E[EBatchJ, =l and L>o] to exclude the secret key computations of the leaf node which is now replaced by the root node of the new subtree. We then add d J to account for the secret key computations done by these new J members. The value d is the level of the highest node that has all its descendents departed. Instead of computing the expected value of d, we can find the upper bound value of d, which occurs when the leaving leaf nodes are evenly distributed in the key tree. Thus, d is given by

Interval-based Distributed Rekeying Algorithms

- We can reduce one rekeying operation if we can simply replace M_5 by M_8 at node 12.
- Interval-based rekeying is proposed such that rekeying is performed on a batch of join and leave requests at regular rekeying intervals. This improves the system performance.
- We propose three interval-based rekeying algorithms, namely Rebuild, Batch and Queue-batch.
- Sponsors are elected at every rekeying event. They coordinate with each other in broadcasting new blinded keys.

RebuildAlgorithm



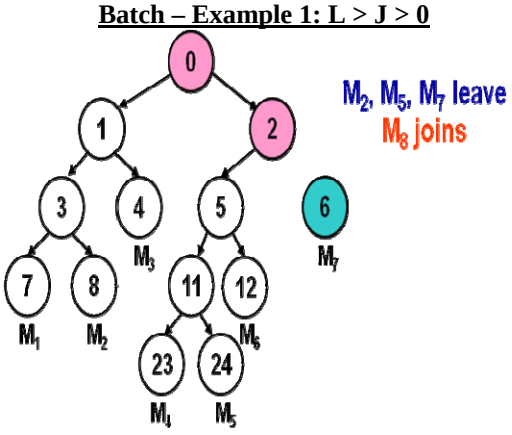
- Intuition: Minimize the height of the key tree so that every member manages fewer renewed nodes in the subsequent rekeying operations.
- Basic Idea: Reconstruct the whole key tree to form a complete tree.
- We can explore the situations where Rebuild is applicable

Batch Algorithm:

- Intuition: Add the joining members to suitable positions.
- Basic Idea:
  - Replace the leaving members with the joining members.
  - Attach the joining members to the shallowest positions.
  - Keep the key tree balanced.

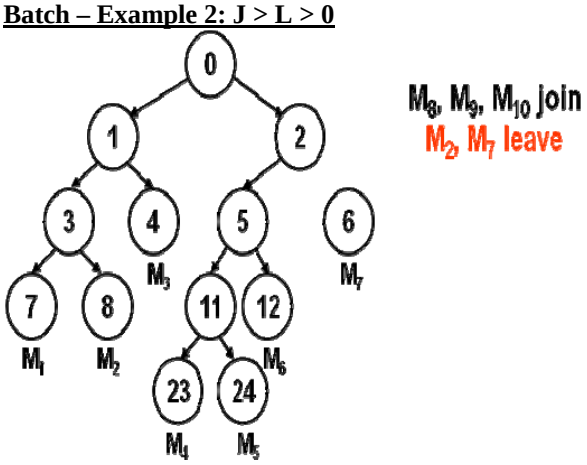
n Elect the sponsors who help broadcast new blinded keys**.**

Batch – Example 1: L > J > 0



- M_8 broadcasts its join request, including its blinded key.
- M_1 rekeys secret keys K_1 and K_0. M_4 rekeys K_5, K_2 and K_0.
- M_1 broadcasts BK_1. M_4 broadcasts BK_5 and BK_2

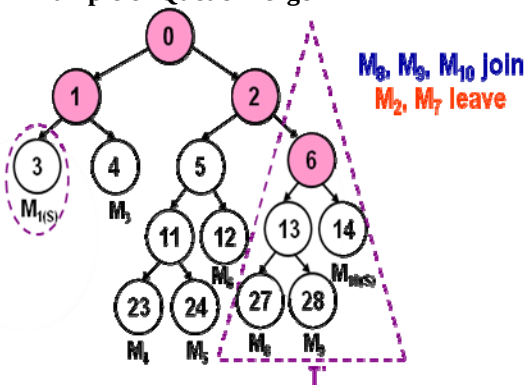
Batch – Example 2: J > L > 0



- M_8 and M_9 form a subtree T_1'. M_{10} itself forms a subtree T_2'.
- M_8 and M_9 compute K_6, and one of them broadcasts BK_6.
- M_1 rekeys K_3 and K_1. M_6 rekeys K_2.
- M_1 broadcasts BK_3 and BK_1. M_6 broadcasts BK_2.

Queue-batch

Example of Queue-merge



- T' is attached to node 6.
- M_{10}, the sponsor, will broadcast BK_6.
- M_1 rekeys K_1. M_6 rekeys K_2.

M_1 broadcasts BK_1. M_6 broadcasts BK_2

Performance Evaluation

- Methods: mathematical models + simulation experiments
- Performance Metrics:
  - Number of renewed nodes: This metric provides a measure of the communication cost.
  - Number of exponentiation operations: This metric provides a measure of the computation load.

Settings:

- There is only one group.
- The population size is fixed at 1024 users.

Originally, 512 members are in the group

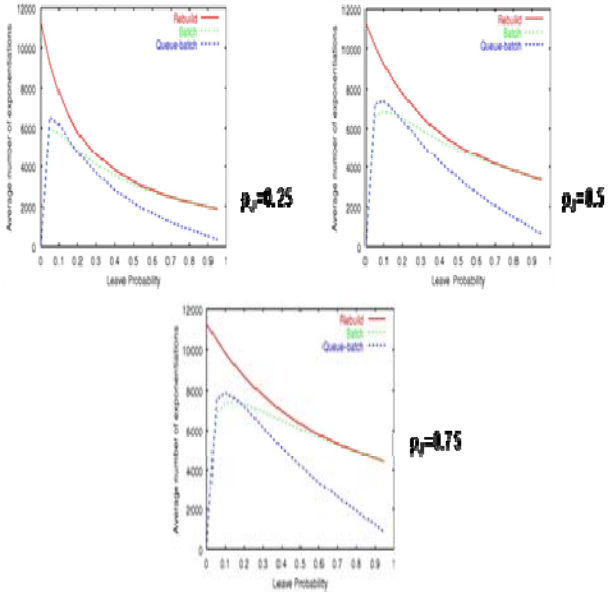
Evaluation 1: Mathematical Models:

- Start with a well-balanced tree with 512 members.
- Obtain the metrics at different numbers of joining and leaving member in a single rekeying interval.
- Queue-batch offers the best performance, and a significant computation/communication reduction when the group is very dynamic.

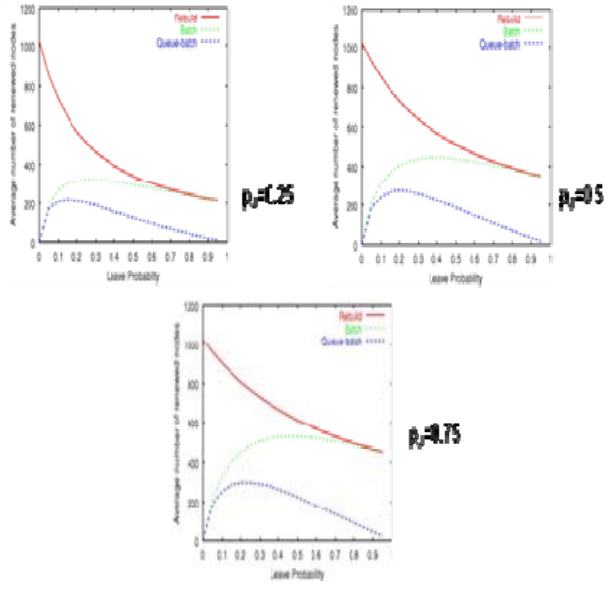
Evaluation 2: Simulation Experiments:

- Start with a well-balanced tree with 512 members.
- Every potential member joins the group with probability p_J, and every existing member leaves the group with probability p_L.
- Evaluate the average / instantaneous metrics at different join/leave probabilities over 300 rekeying intervals.

Average number of exponentiations at different **fixed** join probabilities:



Average number of renewed nodes at different **fixed** join probabilities:



REFERENCE

1] Y. Amir, Y. Kim, C. N. Rotaru, J. L. Schultz, J.Stanton, and G. Tsudik, "Secure group communication using robust contributory key agreement," IEEE Transactions on Parallel and Distributed Systems,vol. 15, no. 5, pp. 468-480, May 2004.

[2] Y. Amir and J. Stanton, The Spread Wide Area Group Communication System, Johns Hopkins University, Baltimore, MD, CNDS-98-4, 1998.

[3] G. Ateniese, M. Steiner, and G. Tsudik, "Authenticated group key agreement and friends," Proceedings of 5th ACM Conference Computer and Communica- tion Security, pp. 17-26, Nov. 1998.

[4] S. Blake-Wilson, and A. Menezes, "Authenticated Diffie-Hellman key agreement protocols," Proceed- ings of 5th Annual Workshop on Selected Areas in Cryptography (SAC' 98), LNCS 1556, pp. 339-361, Springer-Verlag, 1998.

[5] M. Burmester and Y. Desmedt, "A secure and efficient conference key distribution system," Proceed- ings of Advances in Cryptology (Eurocypt'94), LNCS 950, pp. 275-286, Springer-Verlag, 1995.

[6] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Transactions on Information The- ory, vol. 22, no. 6, pp. 644-654, 1976.

[7] A. Fekete, N. Lynch, and A. Shvartsman, "Specifying and using a partionable group communication service," Proceedings of 16th ACM Symp. Principles of Distributed Computing (PODC), pp. 53-62, Aug. 1997.

[8] C. G. G¨unther, "An identity-based key exchange protocol," Proceedings of Advances in Cryptology (Eu- rocypt'89), LNCS 434, pp. 29-37, Springer-Verlag, 1989.

[9] M. Just, and S. Vaudenay, "Authenticated multiparty key agreement," Proceedings of Advances in Cryptology (Asiacrypt'96), LNCS 1163, pp. 36-49, Springer-Verlag, 1996.

[10] Y. Kim, A. Perrig, and G. Tsudik, "Communicationefficient group key agreement," Proceedings of 17th IFIP International Information Security Conference (SEC'01), pp. 229-244, Nov. 2001.

[11] Yongdae Kim, Adrian Perrig, and Gene Tsudik, "Tree-based group key agreement," ACM Transac- tions on Information System Security, vol. 7, no. 1, pp. 60-96, Feb. 2004.

[12] P. P. C. Lee, Distributed and collaborative key agree- ment protocols with authentication and implementa- tion for dynamic peer groups, M. Phil. Thesis, The Chinese University of Hong Kong, June 2003.

[13] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch re-keying for secure group communications," Proceedings of 10th International World Wide Web Conference (WWW'10), pp. 525-534, Orlando, FL, May 2001.

[14] A. Perrig, "Efficient collaborative key management protocols for secure autonomous group communication," Proceedings of International Work- shop on Cryptographic Techniques and E-Commerce (CrypTEC'99), pp.192-202, July 1999.

[15] S. Setia, S.Koussih, and S. Jajodia, "Kronos: A scalable group re-keying approach for secure multicast," Proceedings of IEEE Symposium on Security and Privacy, pp. 215-228, May 2000.

[16] A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," IEEE Transactions on Software Engineering, vol. 29, no. 5, pp. 444-458, May 2003.