Wireless Netw (2008) 14:915–927
DOI 10.1007/s11276-007-0032-y

On optimal batch rekeying for secure group communications in wireless networks

Jin-Hee Cho · Ing-Ray Chen · Mohamed Eltoweissy

Published online: 14 March 2007

Abstract Advances in wireless communications and mobile computing have led to the emergence of group communications and applications over wireless. In many of these group interactions, new members can join and current members can leave at any time, and existing members must communicate securely to achieve application-specific missions or network-specific functionality. Since wireless networks are resource-constrained, a key challenge is to provide secure and efficient group communication mechanisms that satisfy application requirements while minimizing the communication cost. Instead of individual rekeying, i.e., performing a rekey operation right after each join or leave request, periodic batch rekeying has been proposed to alleviate rekeying overhead in resource-constrained wireless networks. In this paper, we propose an analytical model to address the issue of how often batch rekeying should be performed. We propose threshold-based batch rekeying schemes and demonstrate that an optimal rekey interval exists for each scheme. We further compare these schemes to identify the best scheme that can minimize the communication cost of rekeying while satisfying application requirements when given a set of parameter values characterizing the operational and environmental conditions of the system. In a highly dynamic wireless environment in which the system parameter values change at runtime, our work may be used to adapt the rekeying interval accordingly.

J.-H. Cho (✉) · I.-R. Chen · M. Eltoweissy
Department of Computer Science,
Virginia Tech
e-mail: jicho@vt.edu

I.-R. Chen
e-mail: irchen@vt.edu

M. Eltoweissy
e-mail: toweissy@vt.edu

Keywords Secure group communication · Key management · Batch rekeying · Wireless networks · Performance analysis

1 Introduction

Many applications over wired and wireless networks, such as emergency response, mobile commerce, online gaming, and collaborative work, are realized based on a group communication model. Also, wireless networks, such as mobile ad hoc networks and sensor networks, inherently must cooperate to achieve network functionality such as multi-hop routing. Consequently, it is important to assure confidentiality, authenticity, and integrity of messages exchanged among group members which may be end users or network nodes. From this point forward we will use the term "user" to refer to an end-user or a network node.

One way to achieve cost-effective secure group communications is to use a symmetric key, called the *group key*, shared by group members. The group key is distributed by a key server that provides the group key management service. A dedicated key server may be employed, or the functionality may be implemented on the same server offering other services such as authentication. Multiple key servers may co-exist in a clustered network, where a cluster head may play the role of a key server [19]. The group key is employed to encrypt messages sent by a member to the group. Only members of the group are capable of decrypting the messages [2]. In a dynamic group where users may join or leave the group at any time, there are two main security properties commonly associated with rekeying [1, 12]. First, *forward secrecy* assures that an adversary that knows a contiguous subset of old group keys cannot identify subsequent group keys. This guarantees that a member cannot know future

group keys after it leaves the group. Second, *backward secrecy* ensures that an adversary that knows a subset of group keys cannot discover previous group keys. This guarantees that a new member that joins the group cannot learn any previous group keys. To maintain both backward and forward secrecy, the key server needs to perform rekeying (change the group key) when the group membership changes [1, 2].

For large dynamic groups, a join or leave request can occur very frequently. *Individual rekeying* performs a rekey operation whenever a new user joins the group or a current member leaves the group or is evicted. This is not scalable to a large dynamic group because of the significant communication overhead incurred by frequent rekeying in bandwidth-constrained wireless communications. The overhead is exacerbated by the need to authenticate each rekeying message. Moreover, synchronization is difficult to maintain if the group key is rekeyed after each join or leave [2]. To remedy this, researchers have proposed *periodic batch rekeying* [2, 4, 7, 14] by which join and leave requests are aggregated and rekeying is done only periodically. A consequence of batch rekeying is that members may not immediately join or leave the group. Thus, forward and backward secrecy requirements may not be strictly satisfied.

Hardjono et al. [14] proposed *periodic batch rekeying* to decrease rekeying overheads in dynamic group communications. Li et al. [2] proposed the use of periodic batch rekeying to improve efficiency and reduce the out-of-sync problem. Setia et al. [4] described an approach for scalable group rekeying for secure multicast using periodic group rekeying, called *Kronos*. They discussed the inefficiency of individual rekeying under dynamic and large networks, and compared the performance of *Kronos* with other key management protocols using simulation. Yang et al. [7] designed a batch-rekeying algorithm, called *keygem*, to improve scalability and performance of a large and dynamic group. Moharrum et al. [13] proposed a method to handle group dynamics in a multicast key tree and maintain a balanced tree with minimal cost. Recently, Lazos and Poovendran [16] proposed the use of location-aware batch rekeying of key hierarchies in wireless ad hoc networks. Di Pietro et al. [17] proposed *LKH* + + which extends the basic logical key hierarchy (*LKH*) scheme [3] to efficiently provide secure group key management for mobile users. Di Pietro et al. [18] also proposed *LKHW*, a scheme that combined directed-diffusion and *LKH* for efficient key management in wireless sensor networks. Several other schemes as reported in [20] used enhancements of *LKH* for key management in wireless networks.

Based on the literature reviewed, even though *LKH* has been used for key management in wireless networks and periodic batch rekeying has been proposed as an efficient strategy to reduce rekeying overhead by trading off secrecy violation for reduced rekeying overhead, the issue of an optimal batch rekey interval has not been addressed and the relationship between the optimal batch rekey interval and environmental conditions (i.e. the arrival rate of join or leave requests or the probability of trustworthiness in receiving requests) remains to be investigated.

In this paper we propose new threshold-based schemes for periodic batch rekeying and demonstrate that there exists an optimal batch rekey interval for each scheme when given a set of parameter values characterizing the environmental conditions, such as the arrival ratio of join or leave requests or the probability of trustworthiness in the network. A preliminary version of the paper appears in [21]. We compare these threshold-based batch rekey schemes under the same set of environmental conditions, and identify conditions under which a scheme would perform the best in terms of the minimum communication overhead per join/leave operation without violating constraints in secrecy and delay. To reveal the batch rekey interval that optimally explores the tradeoff between acceptable secrecy violation and rekeying delay in wireless networks, we develop a Stochastic Petri net (*SPN*) model to measure and analyze performance metrics, including the communication overhead per operation, probability of secrecy violation, rekeying delay, and batch rekey interval.

This paper has three contributions. First, it develops new threshold-based batch rekeying schemes and identifies an optimal rekey interval for each scheme that would minimize the communication cost per join/leave operation while satisfying secrecy and delay constraints. Second, since periodic batch rekeying can be efficiently used in resource-constrained networks as well as dynamically changing networks, finding an optimal batch rekey interval can be usefully employed in wireless networks. Third, the development of *SPN* models to measure performance metrics for finding an optimal batch rekey interval is a novel approach in this field. The reason we choose *SPN* as our analysis tool is that *SPN* provides concise representation of underlying semi-Markov/Markov model and is capable of dealing with a large number of states. Furthermore, solution techniques in *SPN* can consider general time distributions by using *SPNP* (SPN Package) [9], thus allowing an *SPN* model to be easily defined and constructed.

The rest of the paper is organized as follows. Section 2 states the system model and assumptions used in the paper. Section 3 describes threshold-based batch rekeying algorithms developed with the objective to minimize the communication overhead per group join/leave operation while satisfying secrecy and delay requirements for wireless group communication systems. Section 4 develops mathematical models for performance analysis of the proposed threshold-based schemes. Section 5 presents and analyzes numerical results obtained from applying the mathematical models. Optimal batch intervals of the proposed threshold-based schemes under which the communication cost per join/leave operation is minimized while satisfying delay and secrecy requirements are identified for performance comparison.

Physical interpretations are given to explain the results. Finally, Section 6 concludes this paper and suggests future work.

2 System model and assumptions

We assume that the group communication system is in a wireless environment in which there is a central key distribution server that can authenticate and authorize individual group members. A new member (called a receiver) contacts the key server to request the group key. The key server authenticates the receiver with a standard authentication protocol and establishes a secure channel that provides confidentiality, integrity, and authenticity. If the receiver is authorized, the key server sends group key information to the receiver. A group member encrypts messages with the group key to accomplish confidentiality and limit access to authorized receivers.

We consider that the central key server maintains a key tree based on the *Logical Key Hierarchy* (*LKH*) key distribution protocol [3] to efficiently update the group key after a join or leave event to satisfy *forward* and *backward secrecy* requirements. Each node in the tree indicates a cryptographic symmetric key. The key distribution center connects each group member with one leaf node of the tree and the following invariant will be always maintained: each group member knows all the keys from its leaf node up to the root node, but no other key in the key tree. We call the set of keys that a member knows the *key path*. Since all members know the key at the root node, that key plays a role as the group key. For instance, the key path for member M_2 in Fig. 1 is composed of the keys K_5, K_2, and K_1. When a new member joins the group, the key server sends it all the keys on the key path over a secure channel. When a member leaves the group, the key server needs to update all the keys that the member knows, that is, all the keys on the key path. The main reason for utilizing such a key tree is to efficiently update the group key when join and leave events occur.

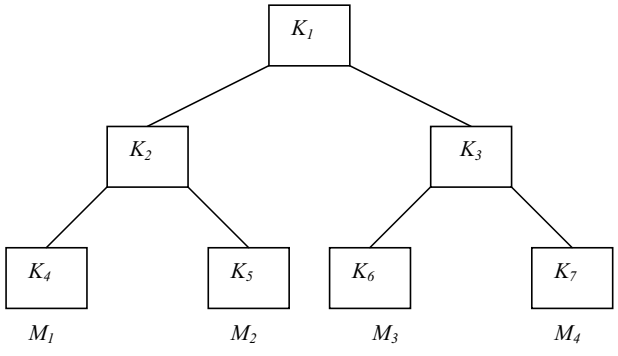


Fig. 1 An example hierarchical key tree

Note that the key update after a member leave event only requires a message of length $2klog_2(N)$ bits (where k is the length of a key, and N is the number of members). Also, a key update operation after a new member join event requires a message of length $k(2log_2(N) - 1)$. One main benefit of *LKH* is that each secure key update only requires a broadcast message size that is logarithmic in the number of group members [3, 16].

We assume that the inter-arrival times of join requests and leave requests are exponentially distributed with rates λ and μ, respectively. This assumption allows us to construct an *SPN* model that can be evaluated using tools such as *SPNP v4* [10]. The assumption of exponential distribution can be relaxed easily by defining other time distributions and evaluating the model using *SPNP v6* [9].

We assume that periodic batch rekeying is employed in the resource-constrained wireless network to alleviate rekeying overheads in terms of the communication overhead incurred due to join or leave requests. We assume that a user cannot join the group unless it is authorized by the authentication server. In this case the join request is identified as a "trusted" join. If a user can join without authentication, then we term that join as "untrusted" join. Untrusted joins are not allowed in our model presented in this paper. A leave request also may be "trusted" or "untrusted." A trusted leave is the one issued by a user that voluntarily leaves the group, for example, because it has moved to another location. On the other hand, a leave is untrusted if the leave is due to the eviction of the group member. If rekeying does not take place immediately after an untrusted leave, a period of security vulnerability occurs until rekeying takes place. When processing a leave operation, the key server is able to differentiate a trusted leave operation from an eviction operation. The probability of trustworthiness for leave operations, P_t, thus can be computed by the key server as the ratio of the number of trusted leave operations over the total number of leave and eviction operations statistically collected by the key server periodically.

3 Threshold-based periodic batch rekeying

A class of batch rekeying schemes is proposed in this paper based on the notion of "thresholds" to govern the maximum number of requests (either join or leave, or both) that can be accumulated in the key server, beyond which rekeying will be performed. As a baseline, we consider a periodic batch rekey algorithm (called *ULT* below) for which only one threshold, say *k3*, is used. When *k3* is reached, rekeying will be performed. We also consider more sophisticated periodic batch rekey schemes for which two thresholds, say *k1* and *k2*, are used and when either *k1* or *k2* has reached, rekeying will be performed. By using thresholds, a threshold-based policy

thus identifies the set of states in which rekeying will be performed, thereby implicitly determining the time interval between two rekeying operations.

The behavior of periodic batch rekeying schemes proposed can be described by a state machine with a 3-component state representation (a, b, c), where a is the number of trusted join requests, b is the number of trusted leave requests, and c is the number of untrusted leave requests, respectively. We consider three different threshold-based batch rekeying schemes as follows:

1. *Untrusted Leave Threshold-based (ULT)*: This scheme has only one threshold, say *k3*, to check against the number of untrusted leave requests (i.e., c in the state representation). This scheme only guards against untrusted leave requests irrespective of the traffic pattern of trusted users. For the special case in which *k3* is 1, *ULT* degenerates to individual rekeying for untrusted requests. We use *ULT* as a baseline scheme against which we compare the performance characteristics of two other more sophisticated batch rekeying schemes described below.
2. *Trusted and Untrusted Double Threshold-based (TAUDT)*: This scheme has two thresholds, *k1* and *k2*, with *k1* checking against the number of trusted requests (i.e., $a+b$) and *k2* checking against the number of untrusted leave requests (i.e., c).
3. *Join and Leave Double Threshold-based (JALDT)*: This scheme has two thresholds, *k1* and *k2*, with *k1* checking against the number of trusted join requests (i.e., a) and *k2* checking against the number of leave requests including both trusted and untrusted leave requests (i.e., $b+c$).

To consider untrusted requests, the probability of trustworthiness (P_t), which indicates the percentage of trusted requests received, is given in all three schemes. For untrusted join requests, the key server does not accept the new node's join request through authentication and authorization. Thus, only untrusted leave requests need to be considered by the key server. In Fig. 2, K_1, K_2, K_3, and K_4 refer to the group keys updated in each interval. Rekeying is performed only at the end of the batch rekey interval defined as the period between two successive group key updates, such as between K_1 and K_2 labeled in Fig. 2.

Two application-specific constraints are considered in this work: probability of secrecy violation (P_v) and delay (D) incurred due to periodic batch rekeying. The delay parameter (D) refers to the average latency experienced per join or leave operation. The probability of secrecy violation (P_v) is measured by the proportion of the time the secrecy requirement is violated. Note that we only need to consider forward secrecy violation (caused by delayed rekeying for leave requests). That is, when a new member joins the group, there is no backward secrecy violation because no key is ever issued to the new member until the end of the batch interval.

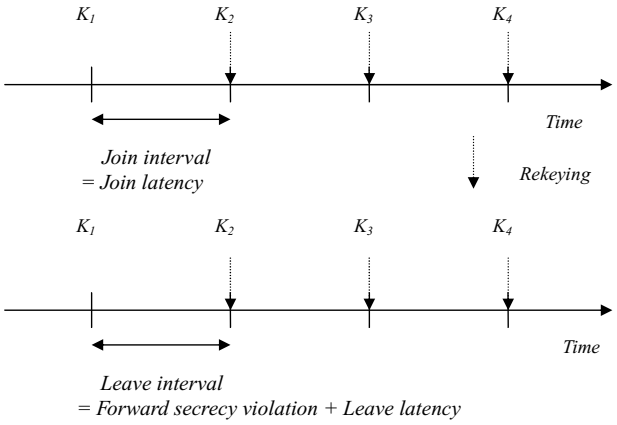


Fig. 2 Periodic batch rekeying with respect to join and leave events

On the other hand, when an untrusted member requests to leave the group, there is a forward secrecy violation since the untrusted member does not leave immediately right after it requests a leave operation, and has to stay until the end of batch rekey interval, allowing it to learn group information. As a result, by the probability of secrecy violation, we refer to the proportion of the time the forward secrecy is violated due to the presence of untrusted users having requested to leave the group.

Note that we do not distinguish join interval from leave interval because join and leave events are aggregated and processed at the end of each batch interval through rekeying. The optimal batch rekey interval (T) is the interval at which the overhead is minimized while satisfying the two application-level constraints in terms of the probability of secrecy violation and delay caused by the postponed rekeying, e.g., 5% of secrecy violation (P_v) and 5 seconds of delays (D).

A simple optimization feature is used to reduce communication overhead taking advantage that the key server in our design has both join and leave requests for rekeying. That is, a new join member can take the place of a leave member in the key tree. Thus, for each pair of join and leave requests, the key server only needs to generate new keys along the paths of the leave members and give the new keys to the new join members. Recall that a state in our design is represented by (a, b, c) where a is the number of trusted join requests, b is the number of trusted leave requests, and c is the number of untrusted leave requests. The key server applies the following procedure when performing a rekeying operation at the end of each batch interval:

- if $a > b+c$, then the server will process $b+c$ join-leave request pairs before processing $a-(b+c)$ join requests;
- if $a = b+c$, then the server will process $b+c$ join-leave request pairs;
- if $a < b+c$, then the server will process a join-leave request pairs before processing $(b+c)$–a leave requests.

4 Performance model

Table 1 summarizes the notation used for parameters in the paper. For *ULT*, we derive analytical closed-form solutions below to calculate the minimum communication overhead per operation (S), the probability of secrecy violation (P_v), and the delay (D) occurred due to periodic batch rekeying.

Let T be the average batch rekey interval in *ULT*, which can be calculated as follows:

$$\frac{1}{\mu(1-P_t)} \times k3 \tag{1}$$

where $\frac{1}{\mu(1-P_t)}$ is the average inter-arrival time of an untrusted leave request.

Thus for *ULT*, at the end of each batch rekeying interval, the state of the system represented by (a, b, c) (see Table 1 for their definitions) will have the following state variable values:

$$a = \lambda \times P_t \times T, \quad b = \mu \times P_t \times T, \quad c = \mu \times (1 - P_t) \times T \tag{2}$$

Consequently, based on the procedure used by *ULT* for performing rekeying at the end of each batch interval, the total communication overhead bits (C_m) in *ULT* can be computed as follows:

$$\begin{aligned}
&\textit{if } a >= (b + c),\\
&\quad \textit{then } J \times (b + c) \times 2\log_2 N + J \times (a - b - c)\\
&\qquad \times (2\log_2 N - 1)\\
&\qquad = J \times a \times 2\log_2 N - J \times (a - b - c)\\
&\textit{else if } a < (b + c),\\
&\quad \textit{then } J \times a \times 2\log_2 N + J \times (b + c - a) \times 2\log_2 N\\
&\qquad = J \times (b + c) \times 2\log_2 N
\end{aligned} \tag{3}$$

Let S_{cm} be the communication overhead (referring to the communication delay) required for performing batch rekeying with the unit of time. Let T_b be the overhead for broadcasting in the wireless network. Then, S_{cm} can be calculated as the sum of T_b and the packet transmission time calculated as the communication overhead bits (C_m) given by Eq. (3) divided by the wireless bandwidth, i.e.,

$$S_{cm} = T_b + \frac{C_m}{BW} \tag{4}$$

Note that T_b includes the wireless channel contention time and the wireless propagation time for broadcasting a message, both of which can be monitored by the key server. In practice, the key server can timestamp every broadcast message prior to transmission, and, based on the timestamps of acknowledgements returned from members, deduce T_b as the average time difference minus the transmission time.

The average communication overhead per join/leave operation (S) in *ULT* for rekeying is simply equal to the total overhead divided by the number of leave/join operations, i.e.,

$$S = \frac{S_{cm}}{(a + b + c)} \tag{5}$$

The probability of secrecy violation (P_v) due to periodic batch rekeying in *ULT* is calculated as the proportion of time in which forward secrecy is violated because of the presence of untrusted leave requests, i.e.,

$$P_V = \frac{\left((k3 - 1)/k3\right) \times T + S_{cm}}{(T + S_{cm})} \tag{6}$$

Table 1 Notation

Symbol	Meaning
λ	Arrival rate of join requests
μ	Arrival rate of leave requests
P_t	Probability of trustworthiness, i.e., probability that a leave request is issued from a trusted user
T_b	Average overhead (delay) for broadcasting in the wireless network due to wireless channel contention and propagation
BW	Network bandwidth (Mbps)
C_m	Communication overhead bits in a batch rekey operation
S_{cm}	Average communication overhead (delay) for batch rekey
S	Average communication overhead (delay) per join/leave
P_v	Average probability of secrecy violation
D	Average delay occurred per join/leave operation
T	Average batch rekey interval
N	Total number of members in the group
J	Length of each key (bits)
a	Number of trusted join requests
b	Number of trusted leave requests
c	Number of untrusted leave requests
ULT	Untrusted Leave Threshold-based: This scheme has only one threshold, *k3*, to check against the number of untrusted leave requests (i.e., c in the state representation)
TAUDT	Trusted and Untrusted Double Threshold-based: This scheme has two thresholds, *k1* and *k2*, with *k1* checking against the number of trusted requests (i.e., $a + b$) and *k2* checking against the number of untrusted leave requests (i.e., c).
JALDT	Join and Leave Double Threshold-based: This scheme has two thresholds, *k1* and *k2*, with *k1* checking against the number of trusted join requests (i.e., a) and *k2* checking against the number of leave requests (i.e., $b + c$)
k1	First threshold used by *TAUDT* and *JALDT*
k2	Second threshold used by *TAUDT* and *JALDT*
k3	Only threshold used by *ULT*

Here $T + S_{cm}$ in the denominator is a base observation period and $[(k3–1)/k3] \times T + S_{cm}$ in the numerator is the duration within the base observation period in which forward secrecy is violated. Note that when $k3 = 1$, the probability of secrecy violation (P_v) is simply $S_{cm}/(T + S_{cm})$ because in this special case an untrusted leave request arrives at the system in every T time interval on average and as soon as it arrives, the system immediately takes S_{cm} to perform rekeying to process the arriving untrusted leave request (because $k3 = 1$), during which forward secrecy is violated.

The delay per join/leave operation (D) in *ULT* is obtained by:

$$D = S + \frac{T}{2} \tag{7}$$

Here $T/2$ is the average wait time for batch rekeying as experienced by an operation and S is the average communication overhead per join/leave operation. Through a sanity check that compares D with the response time per operation, we validate that the calculated D is almost the same as the response time per operation, thus justifying its use. Here the response time, R, is obtained by using *Little's law* $R = n/X$ [11], where n is the average number of requests and X is the throughput of the system.

For *TAUDT* and *JALDT*, there are too many states to yield closed-form analytical expressions. Therefore, an *SPN* model is developed to measure performance metrics including P_v, D, T, and S. Figure 3 shows our *SPN* model. For convenience, Table 2 lists the places, transitions, transition rates, arcs and arc multiplicities used in the *SPN* model.

We first briefly introduce the nomenclature necessary for the comprehension of an *SPN* model [9]. An *SPN* model consists of entities including transitions, places, arcs, and tokens. A token is used as a marker; it can be used to represent a user request. A place is a token place-holder to contain tokens representing join and leave requests; it is normally given a

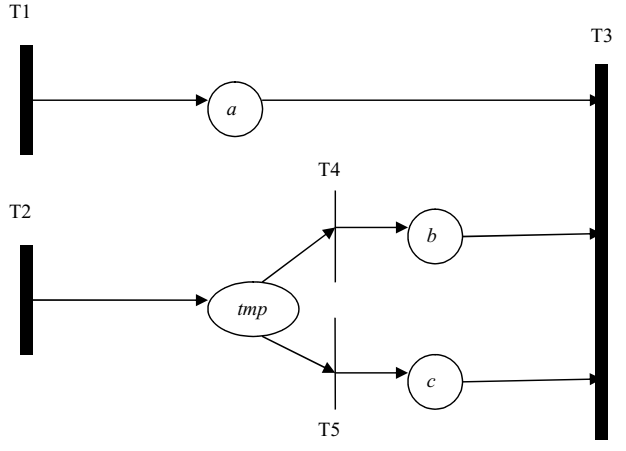


Fig. 3 SPN model for TAUDT and JAUDT

Table 2 Places, transitions, transition rates, arcs and arc multiplicities for the *SPN* model

Place	Meaning
a	$mark(a)$ indicates "a" (number of trusted join requests).
b	$mark(b)$ indicates "b" (number of trusted leave requests).
c	$mark(c)$ indicates "c" (number of untrusted requests).
tmp	$mark(tmp) = 1$ indicates that a leave request has just arrived; $mark(tmp)$ is always 1 or 0.

Transition	Type	Rate or probability
T1	Timed	λP_t
T2	Timed	μ
T3	Timed	$1/S_{cm}$
T4	Immediate	P_t
T5	Immediate	$1 - P_t$

Input arc	Multiplicity	Output arc	Multiplicity
tmp – T4	1	T1– a	1
tmp – T5	1	T2– tmp	1
a – T3	$mark(a)$	T4– b	1
b – T3	$mark(b)$	T5– c	1
c – T3	$mark(c)$		

distinct name that conveys the meaning of a state component, e.g., place a in Fig. 3 holds the number of trusted join requests, place b holds the number of trusted leave requests, and place c holds the number of untrusted leave requests (corresponding to a, b and c in Table 1). The function $mark(p)$ is used to return the current number of tokens held in place p. Typically, state components in the state representation of the underlying Markov or semi-Markov model correspond to places in an *SPN*. Since a state in our model has three components, namely, a, b, and c, three places, namely, a, b and c are created for these state components, respectively. Place tmp is a temporary placeholder, which does not correspond to any state component and is used to hold newly arriving leave requests.

A *transition* represents an event. If a *timed* transition is fired in an *SPN* then it means that an event associated with the transition has occurred, e.g., a leave request arrives after a time exponentially distributed (or generally distributed) has elapsed in the *SPN* model. For modeling convenience, we also allow *immediate* transitions to exist in an *SPN* model. An immediate transition occurs instantaneously without taking any time when the transition fires.

Arcs connect places to transitions. We differentiate input arcs from output arcs. An *input arc* goes from an input place to a transition, while an *output arc* goes from a transition to an output place. An arc can be associated with a multiplicity to indicate the number of tokens associated with the arc; the default is 1 if not specified. A transition can be optionally associated with an *enabling function* to explicitly check

conditions to be satisfied to allow the transition to be fired. An enabling function will return either *true* or *false* depending on the current state of the system. For example, *TAUDT* will perform rekeying when place c holds a number of tokens equal to the *k2* threshold, or places a and b altogether hold a number of tokens equal to the *k1* threshold. In Fig. 3, a transition can fire if the following two conditions are satisfied: (a) there are at least m tokens in each of its input places connected to it by an input arc with multiplicity of m; (b) the associated enabling function (if one is assigned) returns *true*.

Below we explain how the *SPN* model shown in Fig. 3 is constructed:

- When a trusted join request arrives, a token is created to move to place a used to hold the number of trusted join requests. This is modeled by transition *T1* with rate λP_t. Note that untrusted join requests will be detected by the key server, so the transition rate here is λP_t only to account for the arrival rate of trusted join requests. We use a parameter, P_t, to denote the probability of trustworthiness, that is, the probability that a request is issued from a trusted entity.
- When a trusted or untrusted leave request arrives, a token is created to move to *tmp*. This is modeled by transition *T2* with rate μ. Our model distinguishes trusted requests from untrusted requests. If the leave request is from a trusted entity, the token in *tmp* flows to b; otherwise, the leave request is untrusted and the token in *tmp* flows to c. The immediate transition *T4* is associated with probability P_t, while transition *T5* is associated with probability $1 - P_t$. They are fired as soon as its input place, e.g., *tmp*, contains a token, after which the token will be moved from *tmp* immediately to b with probability P_t, and to c with probability $1 - P_t$.
- Under *TAUDT* or *JALDT*, when a rekey condition is satisfied, i.e., when either the *k1* or *k2* threshold is reached, rekeying is performed. This is modeled by associating an enabling function with transition *T3* specifying the rekey condition to be satisfied and firing *T3* when it is so. Based on the threshold control policies, the enabling function of *T3* for *TAUDT* is *if mark*(a) $+$ *mark*(b) $=$ *k1 or mark*(c) $=$ *k2, then return true; otherwise return false.* The enabling function of *T3* for *JALDT* is *if mark(a)* $=$ *k1 or mark(b)* $+$ *mark(c)* $=$ *k2, then return true; otherwise return false*. After a rekeying operation is processed by the key server, all the tokens in a, b, and c (representing the join/leave operations accumulated at the server over the batch interval period) are removed through transition *T3* and the state system goes back to the initial state (0, 0, 0), i.e., *mark*(a) $=$ 0, *mark*(b) $=$ 0 and *mark*(c) $=$ 0.

Table 3 lists the enabling functions associated with transitions in the *SPN* model for the *TAUDT* and *JALDT* schemes, reflecting their respective control behaviors for

Table 3 Transitions and associated enabling functions in the *SPN* model

Transition	Enabling function
T1	
TAUDT	*If mark (a) + mark(b) < k1 and mark(tmp) = 0, then return true; otherwise return false.*
JALDT	*If mark (a) < k1 and mark(tmp) = 0, then return true; otherwise return false.*
T2	
TAUDT	*If mark (c) < k2 and mark(tmp) = 0, then return true; otherwise return false.*
JALDT	*If mark(b) + mark (c) < k2 and mark(tmp) = 0, then return true; otherwise return false.*
T3	
TAUDT	*If mark(a) + mark(b) = k1 or mark(c) = k2, then return true; otherwise return false.*
JALDT	*If mark(a) = k1 or mark(b) + mark(c) = k2, then return true; otherwise return false.*
T4	
TAUDT	*If mark (a) + mark(b) < k1 and mark(tmp) = 1, then return true; otherwise, return false.*
JALDT	*If mark (a) < k1 and mark(tmp) = 1, then return true; otherwise return false.*
T5	
TAUDT	*If mark (c) < k2 and mark(tmp) = 1, then return true; otherwise return false.*
JALDT	*If mark(b) + mark (c) < k2 and mark(tmp) = 1, then return true; otherwise return false*

firing the transitions. The average communication overhead per operation (S) is obtained by assigning a reward of $\frac{S_{cm}}{(mark(a)+mark(b)+mark(c))}$ to each rekeying state in which the enabling function of *T3* returns *true*, where S_{cm} is calculated by Eq. (4) whose value depends on the values of a, b, and c in each rekeying state. Specifically, the following formula is used to calculate S in the *SPN* model:

$$S = \sum_{i \in R}^{all} P(i) \times \frac{S_{cm}}{(mark(a) + mark(b) + mark(c))} \tag{8}$$

Here R denotes the set of rekeying states and $P(i)$ denotes the steady-state probability of the system being in state i, which we could easily obtain by evaluating the *SPN* model using *SPNP* [9].

Under *TAUDT* and *JALDT*, secrecy is violated when there is at least one untrusted leave request in the system. Thus,

the violation probability P_v is obtained by assigning a reward of 1 when $mark(c) > 0$, calculated as follows:

$$P_v = \sum_{i \in V}^{all} P(i) \times r_i \tag{9}$$

Here V denotes the set of states in which $mark(c) > 0$, r_i is 1, and $P(i)$ is the probability that the system is in state i in the steady-state.

In order to obtain the average batch rekeying interval T under *TAUDT* and *JALDT*, we transform the *SPN* model shown in Fig. 3 into one in which all rekeying states become absorbing states. Then, in this transformed *SPN* model with absorbing states, by assigning a reward of 1 to all states except the absorbing states, T can be computed by the expected cumulative reward until absorption, $E[Y(\infty)]$, since this *mean time to absorption* corresponds to the average time it takes to reach an absorption state in which rekeying will be performed. Specifically, in the transformed *SPN* model with rekeying states as the absorbing states, T is calculated as follows:

$$T = E[Y(\infty)] = \sum_{i \in S}^{\infty} r_i \int_0^{\infty} P_i(t) dt \tag{10}$$

Here S denotes the set of all states except the absorbing states in the transformed *SPN* model, $r_i = 1$, and $P_i(t)$ is the probability of state i at time t. An *SPN* evaluation tool such as *SPNP* [9] can be readily applied to compute T based on Eq. (10). Once S and T are obtained from Eqs. (8) and (10), the average delay per operation D can be calculated based on Eq. (7) for *TAUDT* and *JALDT*.

5 Numerical results and analysis

This section presents and analyzes numerical results obtained from applying the mathematical models developed for *ULT*, *TAUDT* and *JALDT*. In all cases presented, the number of members in the group (N) is set to 1024 (representing a large dynamic group), the length of each key (J) is 64 bits, $T_b = 5$ msec, and the bandwidth (BW) is 1 Mbps. Changing these parameter values will affect the scale of the results but does not affect the trend. On the other hand, we change the values of other key parameters including the ratio of the arrival rate of join requests to the arrival rate of leave requests (λ: μ) and the probability of trustworthiness (P_t) to see their effects on the results.

We organize the presentation as follows. First, we show that for each of the three batch rekeying algorithms proposed (*ULT*, *TAUDT*, and *JALDT*) an optimal batch rekeying interval (T) exists that would minimize the cost per join/leave operation (S) while satisfying the requirement constraints in terms of delay (D) and secrecy violation (P_v) in Sections 5.1, 5.2 and 5.3, respectively. Then in Section 5.4 we compare these threshold-based schemes head-to-head under identical system conditions characterized by the probability of trustworthiness (P_t) and the ratio of λ: μ, and identify the scheme that performs the best that minimizes S among all. We used a log scale (*base 10*) to represent the values measured.

5.1 Untrusted leave threshold-based (*ULT*) batch rekeying

Recall that the *ULT* batch rekeying scheme is our baseline scheme which *TAUDT* and *JALDT* will be compared against. It only has one threshold, *k3*, to guard the number of untrusted leave requests (i.e., c).

Figure 4 shows the effect of varying *k3* on the probability of secrecy violation, P_v, in *ULT* while setting λ: $\mu = 1$: 0.5 and $P_t = 0.9$. Other λ: μ and P_t values exhibit similar trends and are not shown here. As we can see, P_v increases as *k3* increases. The reason is that *k3* checks against the number of untrusted leave requests (c) in the key server. Therefore, increasing *k3* means that there are more untrusted leave requests in the key server accumulated until rekeying is performed, thus resulting in a higher probability of secrecy violation. Note that when $k3 = 1$, P_v is 0. That means that as soon as the key server accepts an untrusted leave request, it performs a rekey operation immediately, in which case there is no secrecy violation and forward secrecy is preserved without any violation at the expense of performance degradation.

Figure 5 shows the effect of changing *k3* on the delay (D) incurred due to periodic batch rekeying in *ULT*. As shown in Fig. 5, D increases as *k3* grows. The reason is that when a higher threshold (*k3*) is applied for batch rekeying, it takes more time to accumulate the number of untrusted leave requests by the key server to reach the threshold, thus increasing D.

Figure 6 shows the average communication overhead per join/leave operation (S) as *k3* increases. As expected,

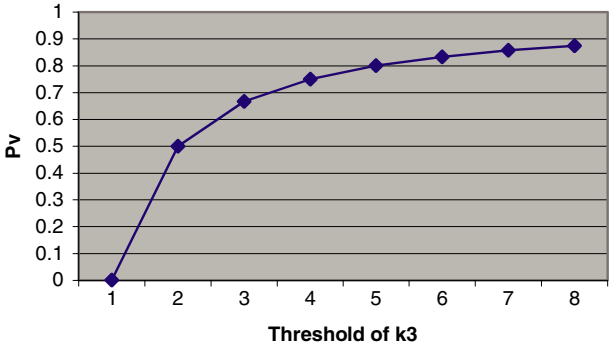


Fig. 4 P_v vs. *k3* under the *ULT* scheme

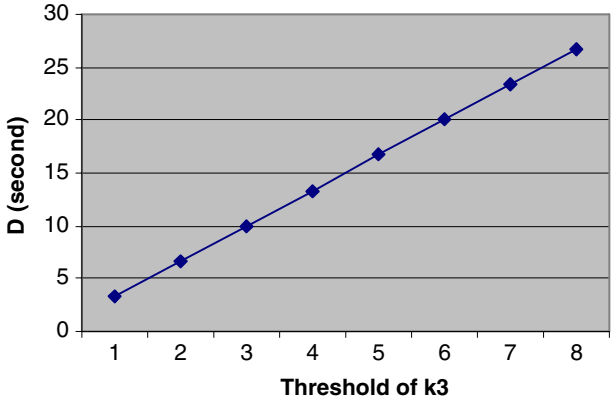


Fig. 5 *D* vs. *k3* under the *ULT* Scheme

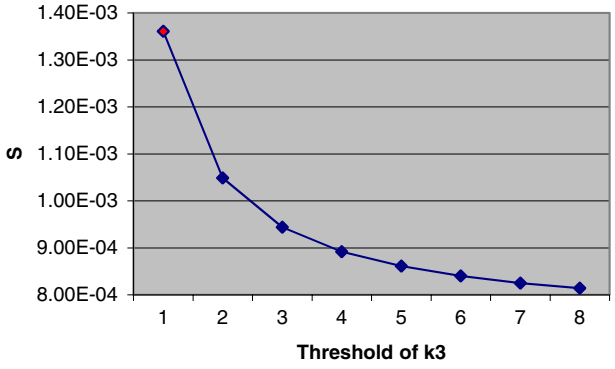


Fig. 6 *S* vs. *k3* under the *ULT* scheme

S decreases as *k3* increases. The optimal *k3* value that minimizes *S* while satisfying the imposed constraints on *D* and P_v, however, is not infinity. For example, when $D = 5$ s and $P_v = 5\%$, *k3* is 1. The corresponding optimal batch rekey interval (T) that minimizes *S* while satisfying *D* and P_v in this case can be readily calculated as 6.67 s based on Eq. (1).

5.2 Trusted and untrusted double threshold-based (*TAUDT*) batch rekeying

Recall that *TAUDT* has two thresholds, *k1* and *k2*, with *k1* guarding against the number of trusted requests $(a + b)$ and *k2* guarding against the number of untrusted requests (c). Figure 7 shows the effect of $(k1, k2)$ on P_v in *TAUDT* with λ: $\mu = 1$: 0.5 and $P_t = 0.9$. As *k1* increases, P_v increases (except when $k2 = 1$ representing the special case that secrecy is perfect) because a higher threshold contributes to more states having violated the secrecy requirement. P_v also increases as *k2* increases in general until *k2* reaches a threshold ($k2 > 2$) beyond which P_v is insensitive to the increase of *k2*. The reason is that with $P_t = 0.9$ most arrivals are trusted requests and thus the effect of *k1* as a threshold dominates the effect of *k2*. We observe that, nevertheless, when P_t decreases, P_v becomes more sensitive to *k2*, and the P_v vs. $(k1, k2)$ curves become more distinct for different *k2* values.

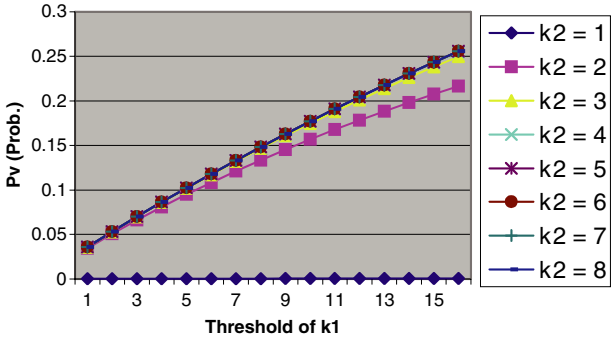


Fig. 7 P_v vs. $(k1, k2)$ under the *TAUDT* scheme

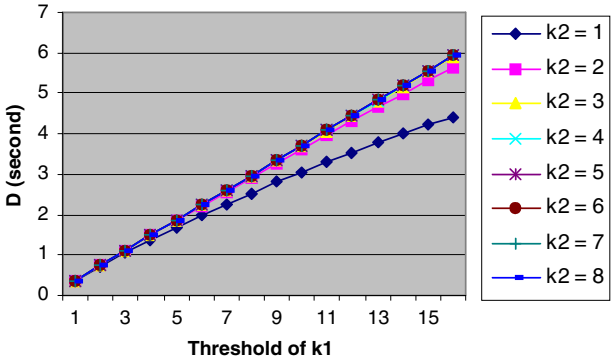


Fig. 8 *D* vs. $(k1, k2)$ under the *TAUDT* scheme

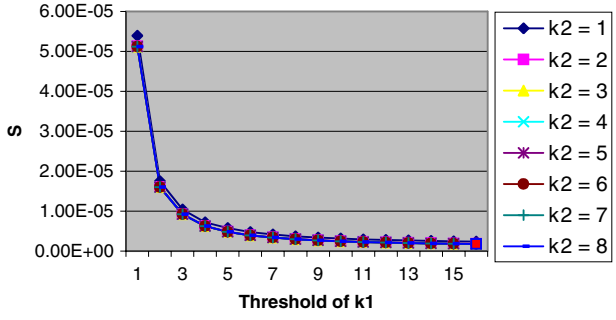


Fig. 9 *S* vs. $(k1, k2)$ under the *TAUDT* scheme

Figure 8 shows *D* vs. $(k1, k2)$. We observe that *D* increases as *k1* increases, because using a higher threshold to aggregate more join or leave requests will result in a higher latency. Here, *D* also increases as *k2* increases although the effect of *k2* is not as significant as *k1* due to a high P_t used. Again, we observe a more significant effect of *k2* on *D* when we decrease P_t.

Lastly Fig. 9 shows the effect of $(k1, k2)$ on the communication overhead per operation (S) in *TAUDT*. As *k1* increases, *S* decreases in the key server because aggregating join and leave events reduces the batch rekeying overhead. Similar to what we have observed in Figs. 7 and 8, since there is a small number of untrusted leave requests (c), *S* is insensitive to increasing *k2*. Figures 7–9 allow us to find the optimal $(k1, k2)$ when given constraints in terms of *D* and P_v. For example,

when $D = 5$ s and $P_v = 5\%$, the optimal setting ($k1$, $k2$) is (16, 1) corresponding to the optimal interval of $T = 8.83$ s. The translation of the optimal ($k1$, $k2$) to the optimal T is through the use of Eq. (10) when evaluating the Petri net model for *TAUDT* discussed earlier.

5.3 Join and leave double threshold-based (*JALDT*) batch rekeying

JALDT has two thresholds, $k1$ and $k2$, with $k1$ checking against the number of join requests (a) and $k2$ checking against the number of leave requests ($b + c$), respectively.

Figure 10 shows the effect of changing $k1$ and $k2$ on P_v in *JALDT*. We see that as either $k1$ and $k2$ increases, P_v increases. The reason is that a higher threshold in either $k1$ or $k2$ brings more states until rekeying is performed, thus contributing to more states in which the secrecy requirement is violated.

Different from the previous results for *TAUDT* (Figs. 7–9), we observe more distinctions between curves as $k2$ increases because $k2$ in the *JALDT* scheme is used as a threshold to check against trusted and untrusted leave requests ($b + c$), not just the number of untrusted requests (c) as in *TAUDT*.

Figure 11 shows the effect of increasing $k1$ and $k2$ on D in *JALDT*. We see that as $k1$ and $k2$ increase, D also increases. Again, using higher thresholds introduces more requests accumulated in the key server until a rekey operation is performed, resulting in the delay being increased. Also, because there are more leave requests governed by $k2$ in the key server, D increases as $k2$ increases.

Figure 12 shows the change of average communication overhead per join/leave operation (S) over increasing $k1$ and $k2$. Again as $k1$ and $k2$ increase, S decreases because aggregating more join and leave events for a batch rekeying operation will amortize the cost per operation.

From Figs. 10–12, we can easily determine the optimal ($k1$, $k2$) that would minimize S while satisfying D and P_v. For example, when $D = 5$ s and $P_v = 5\%$, the optimal setting ($k1, k2$) found is (13, 2) corresponding to the optimal interval of $T = 3.96$ s. The translation of the optimal ($k1$, $k2$) to the optimal T is through the use of Eq. (10) while evaluating the Petri net model for *JALDT*.

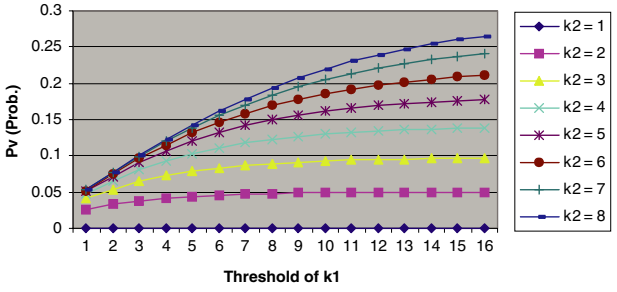


Fig. 10 P_v vs. ($k1$, $k2$) under the *JALDT* scheme

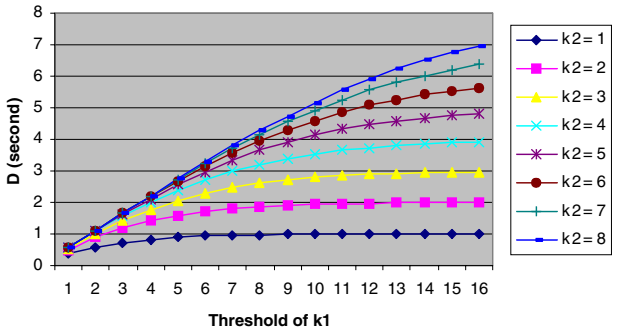


Fig. 11 D vs. ($k1$, $k2$) under the *JALDT* scheme

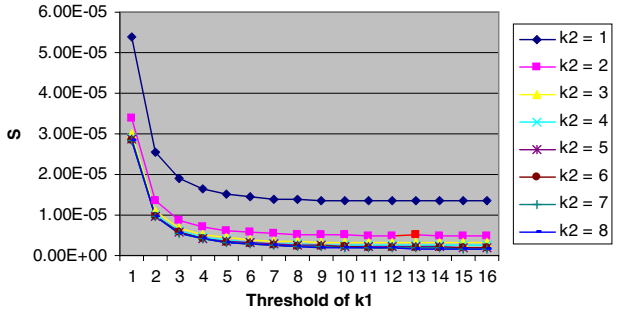


Fig. 12 S vs. ($k1$, $k2$) under the *JALDT* Scheme

5.4 Comparing *ULT*, *TAUDT* and *JALDT* batch rekeying schemes

In this section we compare the minimum S and optimal T obtainable while satisfying imposed constraints on D and P_v by *ULT*, *TAUDT*, and *JALDT*. We test the sensitivity of our results by varying P_t and $\lambda : \mu$ under identical conditions, and identify the best scheme that minimizes S among all while satisfying D and P_v.

Figure 13 shows the "minimum" S (i.e., the optimal S) obtained by *ULT*, *TAUDT*, and *JALDT* as a function of P_t. The ratio of of $\lambda : \mu$ is set to 1 : 0.5 to isolate out its effect. Each data point given is the optimal S found satisfying the constraints that $D = 5$ s and $P_v = 5\%$. We see that for each curve, as the probability of trustworthiness (P_t) increases, S decreases. The reason is that a higher P_t implies less untrusted requests and consequently a lower secrecy violation probability P_v. As a result, a higher P_t reduces the average communication cost per operation. Note that there is no data point at $P_t = 0.9$ under *ULT* because too much delay has occurred (higher than 5 s of D) caused by a low arrival rate of $\mu\ (1 - P_t)$ for untrusted leave requests in this case.

Among the three threshold-based batch rekeying schemes, *TAUDT* performs the best in terms of the minimum communication cost per join/leave operation (S), while satisfying the constraints on D and P_v. On the other hand, *ULT* shows the highest minimum S. The reason is that *ULT* tends to sharply increase P_v as $k3$ increases because *ULT* uses only

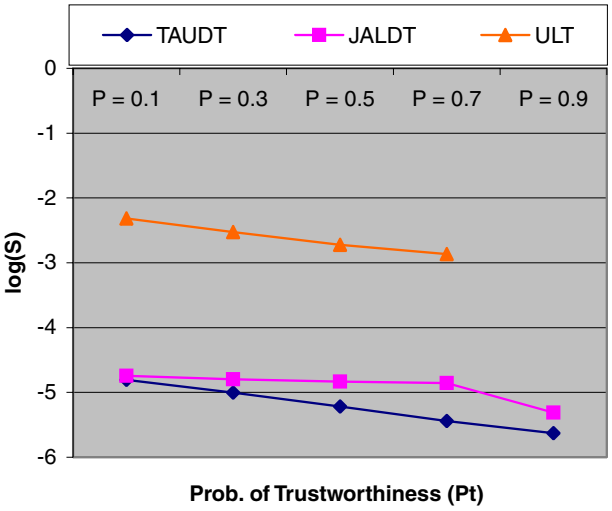


Fig. 13 Comparing *ULT*, *TAUDT*, and *JALDT*: Log (*S*) *vs.* P_t

a single threshold, *k3*, to bound the number of untrusted leave operations. To satisfy the stringent constraint in P_v, *ULT* must select a small *k3* value corresponding to a small optimal *T* and a large minimum *S*. Similarly, since *JALDT* generates a higher P_v than *TAUDT* under identical conditions, *JALDT* has a higher minimum *S* than *TAUDT*. The reason that *JALDT* generates a higher P_v and consequently a higher *S* than *TAUDT* is that *JALDT* has more states having violated forward secrecy because *k2* in *JALDT* checks against $(b + c)$ while *k2* in *TAUDT* only checks against *c*.

Figure 14 shows the optimal *T* corresponding to the minimum *S* found in Fig. 13. Note that the optimal *T* at $P_v = 0.9$ under *ULT* is not available for the same reason that *D* obtained in this case does not satisfy the constraint.

Next we compare *ULT*, *TAUDT*, and *JALDT* as a function of the ratio of λ: μ to test the result sensitivity. Figure 15 shows the minimum *S* obtainable when the arrival rate of leave requests (μ) varies (with the arrival rate fixed at 1) to

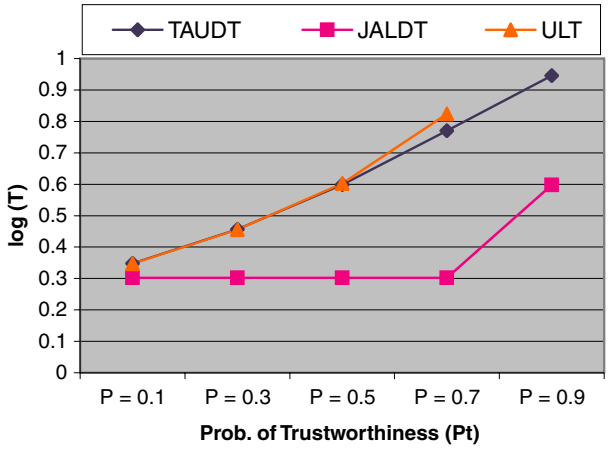


Fig. 14 Comparing *ULT*, *TAUDT*, and *JALDT*: Log (*T*) *vs.* P_t

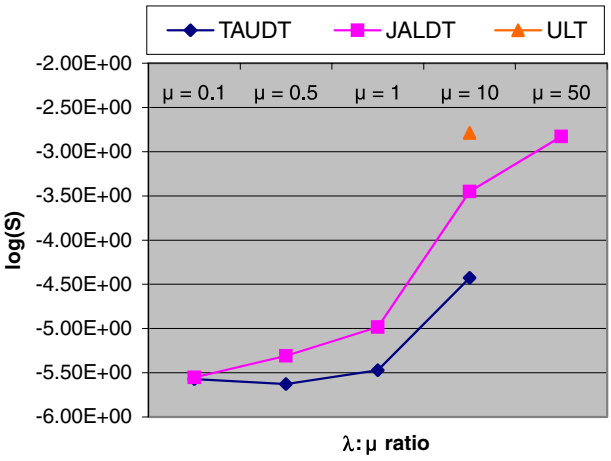


Fig. 15 Comparing *ULT*, *TAUDT*, and *JALDT*: Log (*S*) *vs.* λ: μ

reflect varying ratios of λ: μ. Here we note that the minimum *S* values at $\mu = 0.1, 0.5$, and 1 in *ULT* are not available because *D* obtained exceeds the constraint of 5 s. The minimum *S* at $\mu = 50$ under *ULT* could not be found because P_v obtained is too high due to a very high arrival rate of leave requests. Similarly, the minimum *S* value at $\mu = 50$ under *TAUDT* could not be found because P_v obtained is too high. Figure 15 shows that as μ increases, the minimum *S* also increases. The reason is that a higher μ introduces more untrusted leave requests, resulting in a higher P_v, and thus a higher minimum *S*. As discussed earlier since *TAUDT* is able to generate a lower P_v than *JALDT* under identical conditions, *TAUDT* is able to generate a lower minimum *S*. This is confirmed once again from Fig. 15, which shows that over a wide range of λ: μ ratios, *TAUDT* is the most efficient scheme, showing the lowest minimum *S* among all.

Finally we compare all three threshold-based schemes in terms of the optimal *T* corresponding to the minimum *S* found in Fig. 15. In Fig. 16, as μ increases, the optimal *T* decreases. Comparing Fig. 16 with Fig. 15, whenever there is a minimum *S*, correspondingly there is an optimal

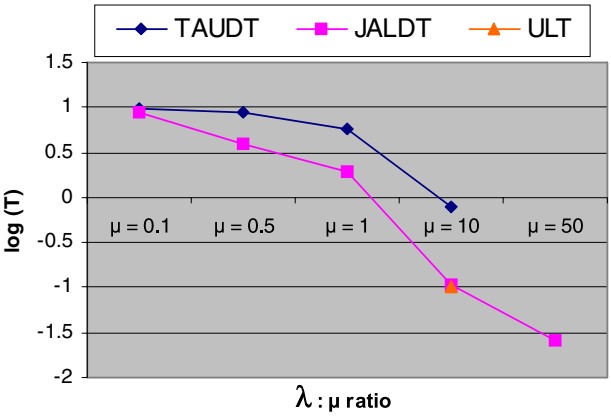


Fig. 16 Comparing *ULT*, *TAUDT*, and *JALDT*: Log (*T*) vs. λ: μ

T generated. We observe that Fig. 16 correlates well with Fig. 15 in terms of the trend shown. Specifically, *TAUDT* has the highest optimal T, as shown in Fig. 16, as it has the lowest minimum S, as shown in Fig. 15. Further, *JALDT* shows the second highest optimal T, followed by *ULT* which is the last although there is only one value at $\mu = 10$ under *ULT*. Based on Fig. 16, we conclude that *TAUDT* has the longest optimal T compared with the two other threshold-based schemes, by reducing the batch rekeying overhead more efficiently.

6 Conclusions and future work

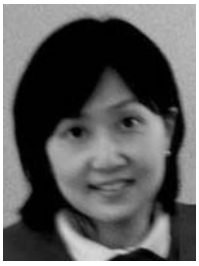
In this work, we have proposed a class of threshold-based batch rekeying schemes with the objective of reducing the communication overhead per join/leave operation (S) while satisfying delay (D) and secrecy (P_v) requirements for wireless group communication systems. We have developed performance models to analyze these batch rekeying schemes, and compare their performance characteristics. We observed that an optimal rekeying interval (T) exists under each of these schemes. Further, by varying the probability of trustworthiness among receiving requests (P_t) and the ratio of the arrival rate of join requests to the arrival rate of leave requests ($\lambda: \mu$) over a wide range, we concluded that among the threshold-based batch rekeying schemes proposed (*ULT, TAUDT*, and *JALDT*), *TAUDT* is able to produce the minimum S and the maximum T, which makes it the most efficient scheme among all. As P_t increases, we observed a decreasing minimum S and an increasing T. As μ increases, we observed an increasing minimum S, and a decreasing optimal T. These results can be used by system designers to determine the optimal T value that would minimize S under *TAUDT*.

As reliability and availability are vital in wireless networks to deal with unreliable group communications [6–8], the *SPN* model developed in the paper can be augmented to take reliability and availability designs into consideration and analyze their effects on the optimal batch rekeying interval. Future research areas include (a) analyzing the effect of insider attacks and intrusion detection system design on the security and performance properties of group communications in wireless systems; and (b) investigating the issue of optimal batch rekeying for the case in which a group consists of multiple subgroups [22].

References

1. C.K. Wong, M. Gouda and S.S. Lam, Secure group communications using key graphs, IEEE/ACM Transactions on Networking 8(1) (Feb. 2000), 16–30.
2. X. Li, Y.R. Yang, M.G. Gouda and S.S. Lam, Batch rekeying for secure group communications, in: *Proceedings of the Tenth International Conference on World Wide Web* Hong Kong (July 2001), pp. 525–534.
3. A. Perrig and J.D. Tygar, *Secure Broadcast Communication in Wired and Wireless Networks* (Kluwer Academic Publishers, 2003).
4. S. Setia, S. Koussih, S. Jajodia and E. Harder, Kronos: A scalable group rekeying approach for secure multicast, in: IEEE Symposium on Security and Privacy (Oakland, CA, May 2000), pp. 215–228.
5. S. Zhu, S. Setia and S. Jajodia, Performance optimizations for group key management schemes, in: *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems* (Providence, Rhode Island, May 2003), pp. 163–171.
6. S. Zhu and S. Jajodia, A scalable and reliable key distribution protocol for multicast group rekeying, Technical Report, GMU, (Jan. 2002).
7. Y.R. Yang, X. Li, X. Zhang and S.S. Lam, Reliable group rekeying: A performance analysis, in: *ACM SIGCOMM 2001* (San Diego, August 2001), Vol. 31, No. 4, pp. 27–38.
8. C.K. Wong and S.S. Lam, Keystone: A group key management system, in: *International Conference on Telecom's*, Acapulco, Mexico (May 2000).
9. G. Ciardo, R.M. Fricks, J.K. Muppala and K.S. Trivedi, *SPNP Users Manual Version 6* (Department Electrical Engineering, Duke University, 1999).
10. G. Ciardo, R.M. Fricks, J.K. Muppala and K.S. Trivedi, *SPNP Reference Guide Version 4* (Department Electrical Engineering, Duke University, 1994).
11. R.A. Sahner, K.S. Trivedi and A. Puliafito, *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the Sharpe Software Package* (Kluwer Academic, 1996).
12. M. Steiner, G. Tsudik and M. Waidner, Key agreement in dynamic peer groups, IEEE Transactions on Parallel and Distributed Systems 11(8) (August 2000), pp. 769–980.
13. M. Moharrum, R. Mukkamala and M. Eltoweissy, Efficient secure multicast with well-populated multicast key trees, in: *Proceedings of the 10th International Conference on Parallel and Distributed Systems*, IEEE Computer Society (July 2004), pp. 215–222.
14. T. Hardjono, B. Cain and I. Monga, *Intra-Domain Group Key Management Protocol* (Internet Draft, Feb. 1998).
15. D.M. Wallner, E.G. Harder and R.C. Agee, Key Management for Multicast: Issues and Architecture (Internet Draft, Sept. 1998).
16. L. Lazos and R. Poovendran, Energy-aware secure multicast communication in ad hoc networks using geographic location information, *IEEE International Conference on Acoustics Speech and Signal Processing* 4 (April 2003), pp. 201–204.
17. R.D. Pietro, L.V. Mancini and S. Jajodia, Security and middleware services: efficient and secure keys management for wireless mobile communications, in: *Proceedings of the 2nd ACM International Workshop on Principles of Mobile Computing*, Toulouse, France (Oct. 2002), pp. 66–73.
18. R.D. Pietro, L.V. Mancini, Y.W. Law, S. Etalle and P. Havinga, LKHW: A directed diffusion-based secure multicast scheme for wireless sensor networks, in: *Proceedings of the 1st International Workshop on Wireless Security and Privacy*, Kaohsiung, Taiwan (Oct. 2003), pp. 397–406.
19. K Ghumman, M.F. Younis and M. Eltoweissy, Location-aware combinatorial key management scheme for clustered sensor networks, IEEE Transactions on Parallel and Distributed Systems 17(8) (Aug. 2006), pp. 865–882.
20. A. Ghosh and F. Anjum, Wireless network security II: last hop topology sensitive multicasting key management, in: *Proceedings of the 1st ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks*, Montreal, Quebec, Canada (Oct. 2005), pp. 63–70.

21. J.H. Cho, I.R. Chen and M. Eltoweissy, Optimization of batch rekey interval for secure group communications in wireless networks, in: *2005 IEEE International Conference on Wireless Networks, Communications, and Mobile Computing,* Vol. 1, Maui, Hawaii (July 2005), pp. 522–527.
22. E. Jung, X.-Y. Liu and M.G. Gouda, Key bundles and parcels: secure communication in many groups, in: *Proceedings of the* 5th *International Workshop on Networked Group Communications*, LNCS 2816, ed. B. Stiller, Springer-Verlag, Munich, Germany (Sept. 2003), pp. 119–130.



Jin-Hee Cho received the B.A. degree from Ewha Womans University in Seoul, Korea in 1997, and the M.S. degree in Computer Science from Virginia Polytechnic Institute and State University, USA, in 2004. Since Fall 2004, she has been pursuing her Ph.D. degree in the Department of Computer Science at Virginia Tech, where she is a Graduate Research Assistant in the Mobile Computing Lab. Her research interests include wireless mobile networks, mobile ad hoc networks, sensor networks, secure group communication, network security, and intrusion detection systems.

Ing-Ray Chen received the B.S. degree from the National Taiwan University, Taipei, Taiwan, and the M.S. and Ph.D. degrees in computer science from the University of Houston. He is a full professor in the Department of Computer Science at Virginia Tech. His research interests include mobile computing, pervasive computing, multimedia, distributed systems, real-time intelligent systems, and reliability and performance analysis. Dr. Chen has served on the program committee of numerous conferences, including as program chair for 29th IEEE Annual International Computer Software and Application Conference in 2005, 14th IEEE International Conference on Tools with Artificial Intelligence in 2002, and 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology in 2000. Dr. Chen currently serves as an editor for *Wireless Personal Communications*, *The Computer Journal*, and *International Journal on Artificial Intelligence Tools*. He is a member of the IEEE/CS and ACM.

Mohamed Eltoweissy is an associate professor in The Bradley Department of Electrical and Computer Engineering at Virginia Tech. He also holds a courtesy appointment in the Department of Computer Science. His research interests are in the areas of information assurance and trust, networking and security in large-scale, ubiquitous, unstructured and resource-constrained environments, service-oriented architectures, and group communications. Eltoweissy received his Ph.D. in Computer Science from Old Dominion University in 1993 and his M.S. and B.S in Computer Science and Automatic Control from Alexandria University, Egypt in 1989 and 1986, respectively. He has over 100 publications in archival journals and respected books and conference proceedings. Eltoweissy is a senior member of IEEE and a member of ACM, ACM SIGBED, and ACM SIGSAC. In 2003, Eltoweissy was nominated for the Virginia SCHEV outstanding faculty awards, the highest honor for faculty in Virginia.