ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Modelling IDS Integrated with Batch Rekeying for Dynamic Group Communication Systems in Mobile ad hoc Networks

M.Poongodi*, PradeepKumar.S and L.Manjula
Department of Computer Science, Rajalakshmi Institute of Tech
Chennai,TamilNadu,India
*poongodi.me@hotmail.com
pradeepk.feb@gmail.com
manjulalets@yahoo.com

Abstract: We investigate performance characteristics of secure group communication systems (GCSs) in mobile ad hoc networks that employ intrusion detection techniques for dealing with insider attacks tightly coupled with rekeying techniques for dealing with outsider attacks. The objective is to identify optimal settings including the best intrusion detection interval and the best batch rekey interval under which the system lifetime (mean time to security failure) is maximized while satisfying performance requirements. We develop a mathematical model based on stochastic Petri net (SPN) to analyze tradeoffs between security and performance properties, when given a set of parameter values characterizing operational and environmental conditions of a GCS instrumented with intrusion detection tightly coupled with batch rekeying. We compare our design with a baseline system using intrusion detection integrated with individual rekeying to demonstrate the effectiveness.

Keywords: Mobile, ad hoc, network, Communication, IDS, Integrated

I. INTRODUCTION

Mobile ad hoc networks (MANETs) are known to have high security vulnerability because of open medium, dynamically changing network topology, decentralized decision-making and cooperation, lack of centralized authority, lack of resources in mobile devices, and no clear line of defence [2]. Two types of security threats exist: insider and outsider attacks. To deal with outsider attacks, prevention techniques such as authentication and encryption have been widely used. To deal with insider attacks, intrusion detection systems (IDS) techniques have been developed for detecting compromised nodes and possibly removing suspicious nodes from the group formation for achieving high-survivability [2].

This paper concerns dynamic group communication systems (GCSs) in MANETs where members of a logical group can join and leave the group, and, while they are in the same group, cooperate to accomplish assigned mission tasks, as in military battlefield situations. We consider design options to deal with both insider and outsider attacks to maintain the notion of secure GCSs. The commonly accepted practice for dealing with outsider attacks in the context of secure GCSs is to maintain a secret key, referred to as the group key, among members. The group key may be rekeyed whenever a change of membership event occurs to maintain confidentiality and secrecy.

Various rekeying algorithms for secure GCSs have been investigated widely in the literature [5, 9, 10, 11]. The most primitive form of rekeying is introduced as *individual rekeying* [10]. Batch rekeying and interval-based distributed rekeying algorithms [9] have been proposed for efficient rekeying for dynamic peer groups.

Recently, threshold-based periodic batch rekeying protocols [5] have been proposed for exploring the tradeoff between secrecy and performance of the system with the objective of identifying the best batch rekey interval. This paper extends our prior work in threshold-based periodic batch rekeying algorithms [5] to remove the assumption of a centralized key server to apply to MANETs by using contributory key agreement protocol [11]. While rekeying techniques provide the first line of defence against outsider attacks, a secure, mission-critical GCS application demands the use of IDS techniques against insider attacks to ensure survivability. Developing IDS techniques to deal with insider attacks for secure GCSs in MANETs is relatively unexplored [2, 3, 6]. Further, most previous studies have discussed IDS techniques separately from rekeying techniques.

In this paper, we integrate batch rekeying with IDS in GCSs and analyze the effect of integration in terms of the tradeoff between performance and security properties of the resulting GCS. Our observation is that IDS techniques employed in the context of secure GCSs must be tightly coupled with rekeying techniques. This is because a node having been identified by IDS as suspicious or compromised can be evicted *immediately*, or *eventually*. The former requires the use of individual rekeying, while the latter could utilize batch rekeying for rekeying efficiency. The decision depends on the system's performance, security, vulnerability, and survivability requirements.

Our goal is to quantify the tradeoff between performance and security properties for a GCS that incorporates both IDS and rekeying techniques. We aim to determine the best IDS detection interval as well as the best batch rekey interval under which security is maximized while performance requirements are satisfied. Specifically, we consider *mean time to security failure* (*MTTSF*) as the security metric for secure GCSs, and we consider the service response time as the performance metric. In effect, we design and analyze IDS techniques tightly

coupled with rekeying techniques applicable to secure GCSs with the goal to identify the best way to execute these protocols based on the tradeoff between security vs. performance metrics.

II. BACKGROUND

We consider two types of IDS protocols for GCSs in MANETs: *host-based IDS* vs. *voting-based IDS*. Host-based IDS is well studied in the literature. We propose voting-based IDS with the objective to improve the system survivability against collusion of compromised nodes. In host-based IDS, each node performs local detection to determine if a neighbouring node has been compromised. Standard IDS techniques such as misuse detection (also called signature-based detection) or anomaly detection [8] can be used to implement host-based IDS in each node. Each node evaluates its neighbors based on information collected, mostly route-related and traffic-related information [8]. The effectiveness of IDS techniques being applied (e.g., misuse detection or anomaly detection) for host-based IDS is measured by two parameters, namely, the false negative probability (*p1*) and false positive probability (*p2*).

We propose voting-based IDS for improved robustness against collusion. Under our voting-based IDS scheme, compromised nodes are detected based on majority voting. Specifically, periodically a node, called a target node, would be evaluated by *m* vote -participants dynamically selected. If the majority decided to vote against the target node, then the target node would be evicted from the system. Voting-based IDS extends from the idea of distributed revocation based on majority voting for evicting a target node in the context of sensor networks [4].

We consider the design of periodicity to allow all nodes to be checked periodically for intrusion detection as well as for tolerance of collusion of compromised nodes in MANETs. We characterize voting-based IDS by two parameters, namely, false negative probability (P_{fn}) and false positive probability (P_{fp}). These two parameters are calculated based on (a) the host-based false negative and positive probabilities (*p1* and *p2*); (b) the number of vote-participants (*m*) selected to vote for or against a target node; and (c) an estimate of the current number of compromised nodes which may collude to disrupt the service of the system. In our voting-
based IDS, if the majority of *m* voting-participants (i.e. > *m* / *2*)

Casts negative votes against a target node, the target node is diagnosed as compromised and is labelled “evicted” from the system. Voting-based IDS is entirely distributed and each node determines its vote based on host-based IDS techniques. The voting-based IDS protocol performs this eviction process periodically. At the beginning of a detection interval, each node would be evaluated by *m* vote-participants; votes are distributed and tallied to decide the fate of the target node.

For the selection of *m* vote-participants in voting-based IDS, each node periodically exchanges its routing information, location, and *id* with its neighbouring nodes. With respect to a target node, all neighbour nodes are candidates as vote-participants. A node with the smallest *id* will elect itself as the coordinator, select *m* nodes randomly (including itself), and broadcast this list of *m* selected vote-participants to all group members. After *m* vote-participants for a target node are selected this way, each vote-participant independently votes for or against the target node by disseminating its vote to all group members. Vote authenticity is achieved via preloaded public/private key pairs. All group members know who *m* vote-participants are, and, based on votes received, can determine whether or not a target node is to be evicted. Under batch rekeying, all evicted nodes along with newly join and leave nodes will be processed at the beginning of the next batch interval and a new group key will be generated based on contributory key agreement among current group members.

We consider three rekeying protocols for GCSs in MANETs:

Individual rekeying: A CKA rekeying is performed right after each join/leave/eviction request.

Trusted and Untrusted Double Threshold-based rekeying with CKA (TAUDT-C): A CKA rekeying is performed after a threshold (*k1*, *k2*) is reached, where *k1* is the number of requests from trusted nodes (i.e. trusted join nodes plus trusted leave nodes) and *k2* is the number of requests due to evictions for the nodes detected by IDS as compromised in the system.

Join and Leave Double Threshold-based rekeying with CKA (JALDT-C): A CKA rekeying is performed after a threshold (*k1*, *k2*) is reached, where *k1* is the number of requests from join nodes (i.e. trusted join nodes) and *k2* is the number requests from trusted leave nodes plus forced evictions for the nodes detected by IDS as compromised in the system.

TAUDT- C and *JALDT-C* extend *TAUDT* and *JALDT* developed in [5] by utilizing a *CKA* for distributed control and removing a single point of failure in MANETs. For brevity, we will just call them *TAUDT* and *JALDT* in this paper. Without loss of generality, this paper considers GDH.3 (called GDH for brevity) [11] as the *CKA* protocol for secret key generation. Due to space constraints, the detailed operation of GDH is referred to [11].

III. SYSTEM MODEL

We assume that the GCS is in a wireless MANET environment in which there is no centralized key server. Each node is preloaded with private/public key pairs of all other group members for authentication purposes. The group key is rekeyed by running a *CKA* protocol with no centralized trust entity to generate and disseminate the group key.

We assume that threshold-based periodic batch rekeying is utilized in resource -constrained MANETs to alleviate rekeying overheads in terms of the communication cost incurred due to join/leave/eviction requests. We assume that a user cannot join the group without authorization. Thus, only “trusted” join is allowed. A leave, on the other hand, may be “trusted” or “untrusted.” A leave is trusted if it is issued by a user that voluntarily leaves the group. A leave is untrusted if the leave is caused by eviction of a detected compromised node. If rekeying is not performed immediately after an untrusted leave, the “to be evicted” node may cause harm to the system since it still possesses the group key.

The group members of the proposed GCS in MANETs are

assumed to be spread over a geographical area (*A*). The workload and operational conditions of a GCS in MANETs can be characterized by a set of model parameters. We assume that the inter-arrival times of trusted join and leave requests, and data packets issued by a node for group communication are exponentially distributed with their rates being λ, $\square$ and λ_q respectively. The assumption of exponential distribution can be relaxed since the SPN performance model developed is capable of allowing any general distribution for a transition time. We assume that the time to perform a rekeying operation upon a membership change event (i.e. join or leave) or a forced eviction is measured based on GDH [11] to realize distributed key management in MANETs. We also assume that inside attackers will attempt to compromise nodes with a variable rate depending on the number of compromised nodes in the system. We use the *linear time attacker* function to model the attacker's behaviours, considering the possibility of collusion of compromised nodes. Compromised nodes are periodically detected by IDS with false positive and false negative possibilities. We assume that IDS will perform its function periodically. The detection interval is dynamically adjusted in response to the accumulated number of intrusion incidents that have been detected in the system. Similar to the attacker behaviour model above, we use a *linear periodic detection* function to model IDS detection activities which will increase linearly with the number of compromised nodes detected.

We assume that *view synchrony* is guaranteed [9] in our GCS, which ensures that messages are delivered reliably and in proper order under the same group membership view. We assume that each node has its own IDS preinstalled to perform intrusion detection activities. This host-based IDS is characterized by the false negative probability and false positive probability represented, respectively, by *p1* and *p2*. As a baseline, the system would perform host-based IDS to evict suspicious nodes. To alleviate collusion, the system would further perform voting-based IDS by which a majority of vote-participants (*m*) must agree to evict a target node before the target node is evicted, where the number of vote-participants (*m*) is a system parameter. Voting-based IDS is characterized by the false negative probability (P_{fn}) and false positive probability (P_{fp}) which depend on *p1* and *p2*, respectively, and the number of compromised nodes in the system. We assume that our GCS enters a security failure state when one of the two conditions stated below is true:

a. Condition C1*:* a compromised member, either detected or not, requests and subsequently obtains data using the group key, leading to illegal data leak-out to a compromised node.

b. Condition C2*:* more than 1/3 of member nodes are compromised by IDS. We assume the Byzantine failure model [7] such that when more than 1/3 of member nodes are compromised, the system fails.

After a member node is detected as compromised by IDS, it can still stay in the system if a batch rekeying protocol is used. This may cause system failure based on Condition C1 defined above. After a node is detected as compromised, it will be evicted for security reasons. There is no recovery mechanism to reinstate evicted members. Initially, all nodes are assumed trusted. We use the following two metrics to measure our design:

a. MTTSF (Mean Time to Security Failure): This metric indicates the lifetime of the GCS before it reaches a security failure state. For a secure GCS, a security failure occurs when either C1 or C2 is true. A design goal is to maximize *MTTSF*. We note that the distribution of security failure time and the probability of security breach are also proper security metrics to measure security failure.

b. R (Service Response Time)**:** This metric refers to the average service response time per group communication operation. This metric is affected by the intensity of rekeying, join/leave/eviction, and IDS operations.

IV. PERFORMANCE MODEL

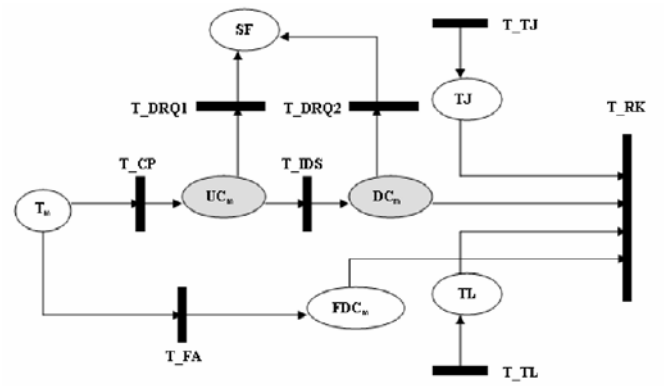

Figure 1: SPN Model.

We develop a mathematical model based on SPN as shown in Figure 1 to describe the behaviours of a GCS instrumented with IDS to cope with insider attacks, as well as batch rekeying to deal with outsider attacks. The goal is to identify optimal settings to maximize *MTTSF* while satisfying imposed performance requirements in terms of *R*.

The SPN model is constructed as follows:

a. We use places to classify nodes. Specifically T_m holds trusted members, UC_m holds compromised nodes that have not been detected by IDS, FDC_m holds nodes falsely diagnosed by IDS as compromised, DC_m holds compromised nodes that have been detected by IDS, *TJ* holds nodes that have issued a join request, and *TL* holds nodes that have issued a leave request.

b. A "token" in our SPN model represents a node in the GCS. The population of each type of nodes is equal to the number of tokens in the corresponding place. For example, $mark(UC_m)$ represents the number of undetected compromised nodes.

c. We use transitions to model events. All transitions in the SPN model are timed transitions. The time taken for a transition to fire depends on the event associated with it. For example, transition *T_RK* stands for a "rekeying" event so the rate at which *T_RK* fires depends on the time taken for the system to perform a rekeying operation based on GDH.

d. We associate triggering conditions with a transition to model conditions under which an event would happen. For example, the triggering condition of *T_RK* depends on the batch rekeying technique used. For *individual rekeying*, if there is a token in FDC_m, DC_m, *TJ*, or *TL*, transition *T_RK* is triggered. For *TAUDT* if either $mark(TJ) + mark(TL)$ reaches *k1* or $mark(FDC_m) + mark(DC_m)$ reaches *k2*,

transition *T_RK* is triggered. For *JALDT* if either *mark(TJ)* reaches *k1* or *mark(TL)+ mark(FDC$_m$) + mark(DC$_m$)* reaches *k2*, *T_RK* fires. Note that places *TJ* and *TL* are used to explicitly count the number of join and leave events to trigger transition *T_RK* according to the threshold-based periodic batch rekeying protocol selected to execute by the system.

e. Initially, all members are trusted; thus, we place all *N* members in place T_m as tokens. Trusted members may become compromised because of insider attacks with a node-compromising rate *A (m$_c$)*. This is modelled by firing transition *T_CP* and moving one token at a time (if it exists) from place T_m to place UC_m. Tokens in place UC_m represent compromised but undetected member nodes.

f. We consider the system as having experienced a security failure when data are leaked out to compromised but undetected members, i.e., due to condition C1. Thus, when a token exists in place UC_m, the system is considered to be in a security vulnerable state. A compromised but undetected member will attempt to compromise data from other members in the group. Because of the use of host-based IDS, a node will reply to such a request only if it could not identify the requesting node as compromised with the per-node false negative probability *p1*. This is modelled by associating transition *T_DRQ1* with rate $p1*\lambda_q$ * *mark (UC$_m$)*. The firing of transition *T_DRQ1* will move a token into place *SF*, at which point we regard the system as having experienced a security failure due to condition C1. Specifically, when *mark(SF)* > 0, the system fails due to condition C1.

g. A compromised node in place UC_m may be detected by IDS before it compromises data in the GCS. The intrusion detection activity of the system is modelled by the detection function with rate $D(m_d)$. Whether the damage has been done by a compromised node before the compromised node is detected depends on the relative magnitude of the node-compromising rate ($A(m_c)$) vs. the IDS detection rate ($D(m_d)$). When transition *T_IDS* fires, a token in place UC_m will be moved to place DC_m, meaning that a compromised, undetected node now becomes detected by IDS. For voting-based IDS, the transition rate of *T_IDS* is $mark(UC_m)*D(m_d)*(1-P_{fn})$, taking into consideration of the false negative probability of voting-based IDS used. Voting-based IDS can also false-positively identify a trusted member node as compromised. This is modeled by moving a trusted member in place T_m to place DC_m after transition *T_FA* fires with rate $mark(T_m)*D(m_d)*P_{fp}$. Note that voting-based IDS parameters, P_{fn} and P_{fp}, can be derived based on *p1* and *p2*, the number of vote-participants (*m*), and the current number of compromised nodes which may collude to disrupt the service of the system. Later we will exemplify how to do the parameterization of P_{fn} and P_{fp}.

h. After a node is detected by IDS as compromised, it is evicted when a rekeying operation is invoked, triggered either by *k1* and *k2* in a double threshold-based periodic batch rekeying protocol. This is modelled by firing transition *T_RK* for evicting detected compromised members. The rate at which transition *T_RK* fires is $1/T_{cm}$. Since an evicted node (in place DC_m) does not leave the group until the next batch rekey interval period, it introduces security vulnerability. We model this data leak-out vulnerability by a transition *T_DRQ2* connecting DC_m and *SF* with rate $p1*\lambda_q$ * *mark (DC$_m$)*. The firing of transition *T_DRQ2* will move a token into place *SF*, at which point we regard the system as having experienced a security failure again due to condition C1. This also models the case that while a double threshold-based periodic batch rekeying algorithm with either *k1* > 1 or *k2* > 1 may improve rekeying efficiency, it may expose the system to this security vulnerability.

i. Join and leave events are modelled by associating transitions *T_TJ*, and *T_TL* with rates of λ and □, respectively.

j. The system is considered as experiencing a security failure if either one of the two security failure conditions, C1 or C2, is met. This is modelled by making the system enter an absorbing state when either C1 or C2 is *true*. In the SPN model, this is achieved by associating every transition in the SPN model with an enabling function that returns *false* (thus disabling the transition from firing) when either C1 or C2 is met, and *true* otherwise. In our model, C1 is *true* when *mark(SF)* > 0 representing that data have been leaked out to compromised members; C2 is *true* when more than 1/3 of member nodes are compromised, where *mark (UC$_m$)+ mark(DC$_m$)* indicates the number of compromised and detected nodes in the system, and *mark(T$_m$) + mark (UC$_m$)+ mark(FDC$_m$)+ mark(DC$_m$)* gives the total number of active nodes in the system.

A. Parameterization:

Here we describe the parameterization process, i.e., how to give model parameters proper values reflecting the operational and environmental conditions of the system.

***a. N*:** This is the number of current active group members in the system, including good nodes in place T_m and undetected compromised bad nodes in place UC_m, so $N = mark(T_m) + mark(UC_m)$. This number evolves dynamically as the system evicts compromised nodes. Since a node leaves the group voluntarily with rate μ and joins the group with rate λ, the probability that a node is active in the group is $\lambda/(\lambda+\mu)$ and the probability that it is not is $\mu/(\lambda+\mu)$. In the SPN model, we initially place $N_{init}\lambda/(\lambda+\mu)$ tokens in place T_m where N_{init} is the initial number of members in the system.

***b.* Λ_J & Λ_L:** These are the aggregate join and leave rates of group nodes for transitions *T_TJ* and *T_TL*, respectively. The aggregate leave rate Λ_L is equal to the number of active group members multiplied by per-node join rate, i.e., $N*\mu$. The aggregate leave rate Λ_L by active members is the same as the aggregate join rate Λ_J by non-active group members.

***c.* T_{cm}:** This is the communication time required for broadcasting a rekey message. The reciprocal of T_{cm} is the rate of transition *T_RK. Based on the GDH protocol T_{cm} can be calculated as:*

Here b_{GDH} is the length of an intermediate value, *H* is a constant representing the number of hops separating any two nodes, and *BW* is the wireless network bandwidth (*Mbps*) in MANETs.

***d. A (m$_c$)*:** This is an attacker function that returns the rate at which a node is compromised in the system. It is also the rate of transition *T_CP*. Among the three different attacker functions proposed in [7], we adopt the *linear time attacker* function $A_{linear}(m_c) = \lambda_c \square m_c$ where λ_c is a base compromising rate and m_c represents the degree of compromised nodes currently in the system, defined by the ratio of *N* to the number of good nodes, i.e., m_c = (*mark*

$(UC_m) + mark(T_m))/ mark(T_m)$.

e. D (m_d): This is a detection function that returns the rate at which intrusion detection is invoked, adjusted based on the accumulated number of nodes that have been detected by IDS. It is also the transition rate of *T_IDS* in our SPN model. We parameterize it based on a *linear periodic detection* function $D_{linear}(m_d) = m_d/T_{IDS,}$ where T_{IDS} is a base intrusion detection interval and m_d represents the "degree" of nodes that have been detected by IDS, calculated by the ratio of N_{init} to N, i.e., $m_d = N_{init} / (mark (UC_m) + mark(T_m))$.

f. ***P_{fn}*** & ***P_{fp}:*** P_{fn} is the probability of false negatives, calculated by the number of compromised nodes incorrectly diagnosed as trusted healthy nodes (i.e. detecting a bad node as a good node) over the number of detected nodes. On the other hand, P_{fp} is the probability of false positives, calculated by the number of normal nodes incorrectly flagged as anomaly over the number of detected normal nodes. We consider intrinsic defect of host-based IDS in each node as well as collusion of compromised nodes in voting-based IDS. For example, a compromised participant can cast a negative vote against a healthy target node and it can cast a positive vote for a malicious node. Equation (2) gives the expressions for computing P_{fn} and P_{fp} as follows:

$$pfp \text{ or } pfn = m - m\left\lceil \frac{m}{2} \right\rceil \sum_{i=0} \left\lceil \frac{c\left\{ \frac{mark(Ucm)}{\lfloor \frac{m}{2} \rfloor + i} \right\} \times c\left\{ \frac{mark(Tm)}{m-(\frac{m}{2}+i)} \right\}}{c(,mark(Tm)+mark(Ucm)} \right\rceil$$

In Equation (2), m is the number of vote-participants with respect to a target node, $mark\ (UC_m)$ is the number of currently compromised nodes and $mark\ (T_m)$ is the number of currently healthy nodes. Nodes that are detected compromised (those in place DC_m) cannot participate in voting-based IDS. Thus, P_{fp} is obtained when the majority of m nodes votes against a good node, including bad nodes who purposefully cast a negative vote against this good node, and good nodes who mistakenly diagnose this good node as a bad node with probability *p2*, resulting in the healthy node being evicted. On the other hand, P_{fn} occurs when the majority of m nodes votes for a bad node, including bad nodes casting a positive vote against this bad node, and good nodes who incorrectly diagnose this bad node as a good node with probability *p1*. Note that p in Equation (2) is *p1* when calculating P_{fn} and is *p2* when calculating P_{fp}.

B. Assessment of Performance Metrics:

a. MTTSF can be obtained by using the concept of *mean time to absorption* (*MTTA*) in the SPN model. The *MTTSF* of the system is simply the expected accumulated reward until any one of the absorbing states is reached.

b. Service Response Time per group communication packet over the system's lifetime, *R*, is calculated by accumulating wireless channel contention delay and transmission delay over *MTTSF* divided by *MTTSF*. We omit the equations for the wireless channel contention delay and transmission delay due to page constraints.

V. NUMERICAL DATA AND ANALYSIS

We present numerical results obtained from evaluating the SPN model developed and provide physical interpretations. Our objective is to identify optimal settings in terms of optimal double thresholds *k1* and *k2* of batch rekeying protocols and optimal intrusion detection intervals that maximize *MTTSF* while satisfying performance requirements in terms of service response time (*R*). In particular, based on the identified optimal *k1* and *k2* thresholds, optimal intrusion detection intervals are identified. We compare the system performance of double threshold-based periodic batch rekeying protocols against the baseline *individual rekeying* integrated with voting-based IDS. We vary design parameters to analyze their effects on system performance. The key default parameter values used are as follows. The join and leave ratio is 4:1 (once per hour and once per 4 hours), the number of vote-participants is m=5, wireless bandwidth is 1 Mbps, the IDS interval varies from 30 to 9600 s, the initial number of members is 60, compromising rate is once per 12 hours, data request rate is once per 30 minutes, and some default parameter values used in calculating *R* are based on [1]. In particular, we use *p1* = *p2* = 1% since in general less than 1% of false positive or false negative rate is deemed acceptance. For voting-based IDS, P_{fn} and P_{fp} are calculated based on Equation (2).

A. Optimal Double Thresholds (k1 and k2):

Figure 2 shows the effect of varying *k1* and *k2* on *MTTSF* for *TAUDT*. We see that when *k2* is 1 (corresponding to immediate eviction without delay), *MTTSF* is way above the other curves, showing a much longer system lifetime among all. The optimal *MTTSF* in *TAUDT* is observed at (*k1*, *k2*) = (4, 1), as shown in Figure 2. We explain why the optimal (*k1*, *k2*) = (4, 1) under *TAUDT* below. Recall that in *TAUDT*, *k1* governs against the number of join/leave nodes ($mark(TJ) + mark(TL)$) while *k2* governs against the number of nodes detected as compromised ($mark(FDC_m) + mark(DC_m)$). As *k2* increases, security failure due to Condition C1 is more likely to occur since a larger *k2* allows more detected compromised nodes to exist. Allowing *k2* larger than 1 significantly deteriorates *MTTSF*. Thus, *k2* is optimized at 1. When *k1*=1, the probability that rekeying is triggered due to *k1* is high. This has the effect of delaying detected compromised nodes (in DC_m) to be removed, which degrades *MTTSF* again due to condition C1. As *k1* increases, the probability that rekeying is triggered due to *k2* increases. This has the effect of quickly removing detected compromised nodes, which increases *MTTSF* as a result. Lastly, as *k1* increases further, not only nodes in DC_m but also nodes in FDC_m are very quickly removed. This has the effect of degrading *MTTSF* due to Condition 2. We also note that when *k2* is greater than 1, there isn't much sensitivity of *MTTSF* on *k2* since *k2* governs untrusted members directly related to security failure.

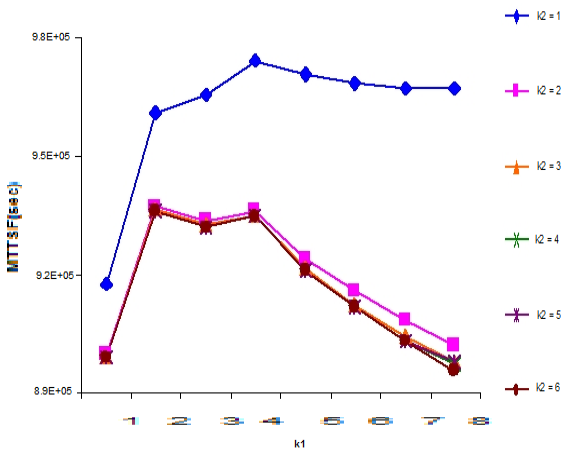

Figure 2: Optimal *k1* and *k2* for *TAUDT* in *MTTSF*.

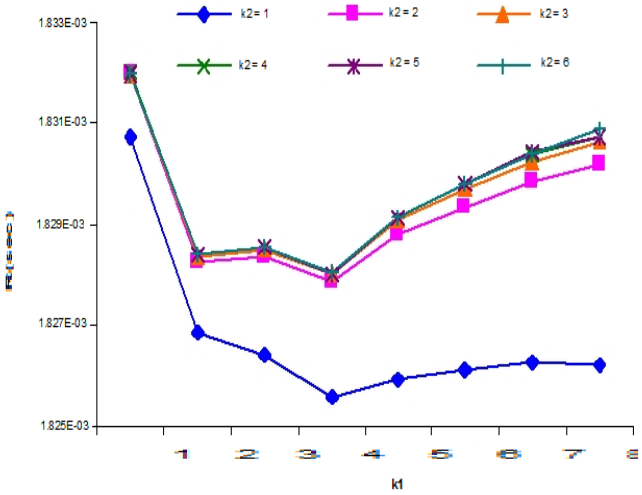

Figure 3: Optimal *k1* and *k2* for *TAUDT* in *R*.

Figure 3 shows the effect of *k1* and *k2* on the service response time, *R*. We observe that the trend shown in Figure 3 strikingly reflects the overall communication cost per time unit vs. *k1* and *k2* (not shown here for brevity) . In Figure 3, we see the optimal

(*k1*, *k2*) at (4, 1) for optimizing *R* matches that for optimizing *MTTSF* in Figure 2. For *JALDT* (not shown here) we have the optimal (*k1*, *k2*) at (5, 2) for both *MTTSF* and *R*. Note that in *JALDT k2* is shared by trusted join and untrusted join requests, so *k2* is more relaxed with a larger threshold compared to *k2* in *TAUDT*.

B. Optimal Intrusion Detection Intervals (T_{IDS}):

Here we analyze the optimal intrusion detection interval (T_{IDS}) based on optimal double thresholds *k1* and *k2* identified, that is, for *TAUDT*, (*k1*, *k2*) = (4, 1) and for *JALDT*, (*k1*, *k2*) = (5, 2) for all T_{IDS} ranges, respectively. We compare system performance under periodic batch rekeying vs. individual rekeying and show that batch rekeying under optimal settings outperforms individual rekeying when IDS is present.

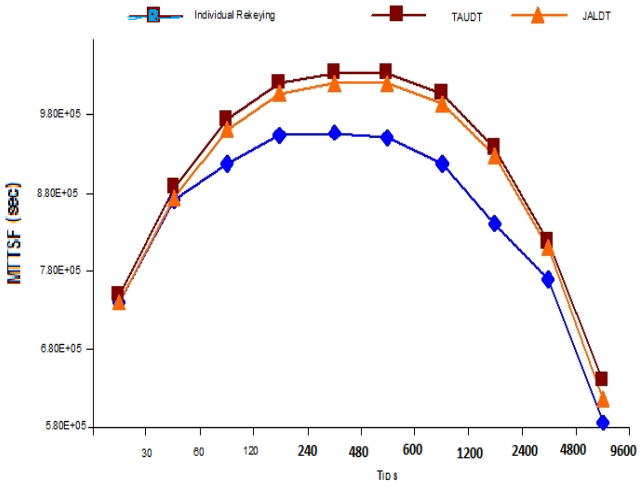

Figure 4: Optimal T_{IDS} in *MTTSF*.

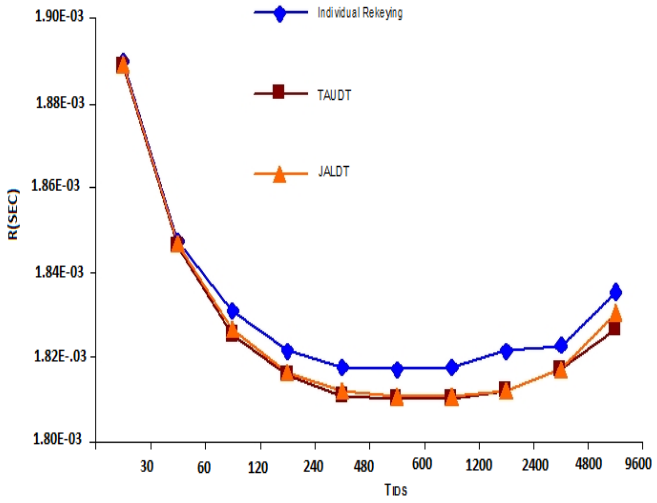

Figure 5: Optimal T_{IDS} in *R*.

Figure 4 shows the effect of three different periodic batch rekeying protocols on *MTTSF* and identifies the optimal intrusion detection interval, T_{IDS}. We observe that there exists an optimal T_{IDS} that maximizes *MTTSF*. In general, as T_{IDS} increases, *MTTSF* increases until its optimal T_{IDS} is reached, and then *MTTSF* decreases after the optimal T_{IDS}. The reason of decreasing *MTTSF* after reaching the optimal point is that the false positive probability (P_{fp}) increases as T_{IDS} decreases, therefore resulting in more nodes being falsely identified as compromised and being evicted from the system. Note that P_{fp} is one aspect of false alarms generated by IDS, so its effect increases when IDS is more frequently triggered. As expected, we observe that the baseline individual rekeying performs the worst, while *TAUDT* performs the best in terms of *MTTSF* among the three. Here *TAUDT* operates at the optimal setting (*k1*, *k2*) = (4, 1) as identified in the paper. On one hand, *k2*=1 allows rekeying to be triggered as soon as possible once a compromised node has been identified for eviction. On the other hand *k1*=4 balances the probability of security failure due to Condition 1 vs. Condition 2, as explained earlier. We note that i*ndividual rekeying* performs the worst because the probability that rekeying is triggered

due to trusted join/leave is high. This has the effect of removing detected compromised nodes in DC_m slowly and decreasing *MTTSF* due to Condition 1 The optimal intrusion detection interval is identified at T_{IDS} = 240 s for individual rekeying, and 480 *s* for *TAUDT* and *JALDT*, as shown in Figure 4.

Figure 5 shows service response time (*R*) vs. intrusion detection interval (T_{IDS}). We again observe that there exists an optimal T_{IDS} that minimizes the service response time in all three curves. The trend can be explained with the same reasoning as for Figure 4. Among three curves in Figure 5, we observe that *individual rekeying* performs the worst while *TAUDT* at the optimal point performs the best. A systems designer can use the results obtained here to identify T_{IDS} that can optimize system performance. To maximize *MTTSF*, T_{IDS} is identified as 480 *s*. To minimize *R*, T_{IDS} is identified as 600 *s*. However, there is an insignificant response time difference between T_{IDS} = 480 *s* and T_{IDS} = 600 *s*. Thus the optimal T_{IDS} in this case is set to 480 *s* that can maximize *MTTSF* while satisfying the service response time (*R*) requirement.

VI. ACKNOWLEDGEMENT

Phu-Gui Feng's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.

VII. REFERENCES

[1]. G. Bianchi, "Performance Analysis of the IEEE 802.11 Distributed Coordination Function," IEEE Journal on Selected Areas in Communications, vol. 18, no. 3, Mar. 2000, pp. 535-547.

[2]. P. Brutch and C. Ko, "Challenges in Intrusion Detection for Wireless Ad-hoc Networks," Proc. Symposium on Applications and the Internet Workshops, 27-31 Jan. 2003, pp. – 373.

[3]. J.B.D. Cabrera, C. Gutierrez, R. K. Mehra, "Infrastructures and Algorithms for Distributed Anomaly-based Intrusion Detection in Mobile Ad-hoc Networks," IEEE Military Communications Conf., MILCOM 2005, vol. 3, 17-20 Oct. 2005, pp. 1831 – 1837.

[4]. H. Chan, V.D. Gligor, A. Perrig, and G. Muralidharan, "On the Distribution and Revocation of Cryptographic Keys in Sensor Networks," IEEE Transactions on Dependable and Secure Computing, vol. 2, no 3, July-Sept. 2005, pp. 233 – 247

[5]. J.H. Cho, I.R. Chen and M. Eltoweissy, "On Optimal Batch Rekeying for Secure Group Communications in Wireless Networks," ACM/Springer Wireless Networks, 2007.

[6]. J.H. Cho and I.R. Chen, "On Design Tradeoffs Between Security and Performance in Wireless Group Communicating Systems," IEEE 1st Workshop on Secure Network Protocols, Boston, Nov. 2005, pp. 13-18.

[7]. F.C. Gärtner, "Byzantine Failures and Security: Arbitrary is not (always) Random," Technical Report IC/2003/20, EPFL, April, 2003.

[8]. Y. Huang and W. Lee, "A Cooperative Intrusion Detection System for Ad Hoc Networks," Proc. 1st ACM Workshop on Security of Ad-hoc and Sensor Networks, Fairfax, Virginia, 2003, pp. 135-147.

[9]. Patrick P.C. Lee, John C.S. Lui, and David K.Y. Yau, "Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups," IEEE/ACM Transactions on Networking, vol. 14, no. 2, April 2006, pp. 263-276.

[10]. X. Li, Y.R. Yang, M.G. Gouda, and S.S. Lam, "Batch Rekeying for Secure Group Communications," Proc. of the Tenth Int'l Conf. on World Wide Web, Hong Kong, July 2001, pp. 525-534.

[11]. M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 8, Aug. 2000, pp. 769-980.