Modeling and Analysis of Intrusion Detection Integrated with Batch Rekeying for Dynamic Group Communication Systems in Mobile Ad Hoc Networks

Jin-Hee Cho
U.S. Army Research Laboratory
Computational and Information Sciences Directorate
jinhee.cho@us.army.mil

Ing-Ray Chen
Virginia Tech
Department of Computer Science
irchen@vt.edu

***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**— Group communication systems, mobile ad hoc networks, batch rekeying, intrusion detection, stochastic Petri net, group key management, security, performance analysis.*

1. 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 defense [2, 23, 33]. 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 [33].

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, so called the group key, among members. The group key may be rekeyed whenever a member joins or leaves (or become evicted). The secret key provides confidentiality and secrecy. Various rekeying algorithms for secure GCSs have been investigated widely in the literature. The most primitive form of rekeying is *individual rekeying* [21, 30], that is, a rekeying operation is performed immediately when a join or leave event occurs. Batch rekeying [12, 24, 25, 31] and interval-based distributed rekeying algorithms [20] have been proposed for efficient rekeying for dynamic peer groups, with the tradeoff of weakening confidentiality as a result of delaying the update of the group key. Recently, threshold-based periodic batch rekeying protocols [6] have been proposed for exploring the tradeoff between secrecy and performance of the system with the objective of identifying the best batch rekey interval to maximize performance while satisfying security properties. This paper extends our prior work in threshold-based periodic batch rekeying algorithms [6] to remove the assumption of a centralized key server to apply to MANETs. We also incorporate contributory key agreement (CKA), i.e., each group member contributes to rekeying of the group key, to deal with group dynamics in a secure GCS setting in MANETs

While rekeying techniques provide the first line of defense against outsider attacks, a secure, mission-critical GCS application demands the use of IDS techniques against insider attacks to ensure survivability. In the literature, IDS techniques for dealing with insider attacks for secure GCSs in MANETs include [2, 3, 10, 13, 15, 22, 27, 28, 29]. However, these IDS techniques have been studied 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. Furthermore, while IDS activities introduce extra communication overhead to detect insider attacks, batch rekeying reduces communication cost by delaying evictions of suspicious members detected by IDS at the risk of exposing the system to security vulnerability.

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 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 per group operation 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. We emphasize that the threshold-based periodic batch rekeying algorithms considered in the paper could degenerate to individual rekeying if the condition dictates that individual rekeying be used to satisfy the security requirement.

This paper has several contributions with respect to GCSs in MANETs. First, we consider the incorporation of security techniques to deal with both outsider and insider attacks to result in secure GCSs in MANETs, i.e., batch rekeying for dealing with outsider attacks and IDS for dealing with insider attacks. Second, we observe and evaluate the tradeoff of security vs. performance properties of the resulting GCS. Third, we perform mathematical analysis based on stochastic Petri net (SPN) to describe the resulting GCS to quantitatively identify optimal settings (i.e., optimal batch rekeying and intrusion detection intervals) that would maximize system lifetime (*MTTSF*) while satisfying performance requirements (i.e., communication latency per operation). The analytical results identified allow the GCS to dynamically determine the best settings to run IDS and rekeying to satisfy the system's performance and security requirements. This work extends from our preliminary work [9] by (a) considering the Group Diffie-Hallman (GDH) algorithm [23] as the CKA protocol for group members to generate and distribute a new group key upon a group membership change event in MANETs; (b) considering "hop-bits" as the communication cost unit for quantifying the network traffic in multi-hop MANETs where information bits may travel through multiple hops to reach the destination; (c) introducing new security and attack models as well as countermeasures to deal with insider and outsider security attacks; (d) introducing new and efficient calculation procedures for obtaining *MTTSF* and the

service response time for performance analysis; and (e) significantly expanding the analysis including analyzing the effects and sensitivity of key parameters on *MTTSF* and the service response time performance metrics.

The rest of this paper is organized as follows. Section 2 describes the background of IDS and threshold-based periodic batch rekeying, as well as contributory key agreement protocols applied for rekeying in this paper. Section 3 gives the system model including assumptions, the attack and security models, and evaluation metrics. Section 4 develops a mathematical model for performance analysis and discusses how model parameter values are given to characterize the operational conditions and how performance/security metrics are calculated. Section 5 analyzes the results obtained from evaluating the mathematical model and identifies optimal settings. Finally, Section 6 concludes the applicability and outlines some future research areas.

2. Background

2.1 IDS Protocols

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 neighboring node has been compromised. Standard IDS techniques such as misuse detection (also called signature-based detection) or anomaly detection [17, 33] 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 [13, 33]. Each node can also *actively* collect IDS information such as recording if a packet sent to a neighbor is not forwarded as requested. A node can collect data either at the MAC layer or application layer [13]. The effectiveness of IDS techniques 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. Our 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 [5] and intrusion tolerance techniques based on secret sharing and threshold cryptography in MANETs [18, 34].

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., $>\lceil m/2 \rceil$) casts negative votes against a target node, the target node is diagnosed as compromised and is labeled "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 neighboring nodes. If a compromised node fakes its id or location, it increases its chance of being detected by host-based IDS preinstalled on each node. With respect to a target node, nodes that are $H_{nb}(m)$-hop away are candidates as vote-participants where $H_{nb}(m)$ is a design parameter. A coordinator is selected randomly so that the adversaries will not have specific targets to launch their attacks. We add randomness to the coordinator selection process by introducing a hashing function that takes in the id of a node- concatenated with the current location of the node as the hash key. The node with the smallest returned hash value would then become the coordinator. Since candidate nodes know each other's id and location, they can independently execute the hash function to determine which node should be the coordinator. The coordinator then selects *m* nodes randomly (including itself), and broadcasts 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.

2.2 Rekeying Protocols

We consider three rekeying protocols for GCSs in MANETs:

- ***Individual rekeying***: A rekeying is performed right after each join/leave/eviction request.
- ***Trusted and Untrusted Double Threshold-based rekeying with CKA (TAUDT-C)***: A 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. That is, when either *k1* or *k2* is reached, a rekeying operation based on *CKA* is performed. This protocol extends *TAUDT* in [6].
- ***Join and Leave Double Threshold-based rekeying with CKA (JALDT-C)***: A 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. This protocol extends *JALDT* in [6].

TAUDT-C is based on separating rekeying operations into "trusted" and "untrusted" groups, whereas *JALDT-C* is based on separating rekeying operations into "join" and "leave" groups. We conceive *TAUDT-C* as the best model to deal with security attacks since it separates untrusted nodes from trusted ones, thus making both thresholds effective. *JALDT-C* can be considered as a baseline model against which *TAUDT-C* is compared. Another possible rekey protocol conceivably is based on three thresholds by separating rekey operations into "join," "trusted leave" and "untrusted leave" groups. We believe it will not be as effective as *TAUDT-C* since it may unnecessarily separate "trusted" operations into two groups, so neither of the two "trusted" thresholds would be effective compared with the "untrusted" threshold. Thus, in this work we will only consider double threshold-based batch rekeying protocols along with individual rekeying. Here we note that *TAUDT-C* and *JALDT-C* extend *TAUDT* and *JALDT* developed in [6] by utilizing a CKA protocol 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) [23] as the CKA protocol for secret key generation. Other than GDH, other distributed CKA protocols such as TDGH [35] and SEGK [36] can be used for implementing rekeying in our approach. Below we briefly explain how GDH works.

$$Stage\ 1: upflow\ \ M_1 \rightarrow M_2 \rightarrow \cdots \rightarrow M_{n-2} \rightarrow M_{n-1}$$

$$message\ size \qquad b_{GDH} \quad b_{GDH} \quad b_{GDH} \quad = b_{GDH}(n-2)$$

$$Stage\ 2: broadcast\ \ M_{n-1} \rightarrow M_i\ \ where\ i \neq n-1$$

$$message\ size \qquad b_{GDH} \qquad = H \times b_{GDH}$$

$$Stage\ 3: response\ \ M_i\ \ \ where\ i \neq n \rightarrow M_n$$

$$message\ size \qquad b_{GDH}\ from\ each\ \ M_i \qquad = H \times b_{GDH}(n-1)$$

$$Stage\ 4: broadcast\ \ M_n \rightarrow M_i\ where\ i \neq n$$

$$message\ size \quad b_{GDH}(n-1)\ intermediate\ values \quad = H \times b_{GDH}(n-1)$$

$$Total\ communication\ cost = nb_{GDH}(2H+1) - b_{GDH}(H+2)$$

Figure 1: Message Size Requirement in GDH.

GDH comprises four stages [25]. Each participant M_i shares a common base α and keeps its secret share N_i. The first stage collects contributions from all group members, M_1, M_2…, M_n. Specifically, M_1 raises α to the power of N_1, performing one exponential computation to generate α^{N1}, M_2 computes $\alpha^{N1\ N2}$ by raising α^{N1} to the power of N_2, and so on until M_{n-1} computes $\alpha^{N1\ldots\ Nn-1}$. After processing the upflow message, M_{n-1} obtains $\alpha^{\prod\{N_k|k\epsilon[1,n-1]\}}$ and broadcasts this value in the second stage to all other participants. In the third stage, every M_i factors out its own exponent and forwards the result to M_n. In the final stage, M_n collects all inputs from all other participants, raises every one of them to the power of N_n and broadcasts the resulting n-1 values to the rest of the group. Every M_i receives this message in the form of $\alpha^{\prod\{N_k|k\epsilon[1,n-1]\ \cap\ k\neq i\}}$ and can easily generate the intended secret key K_n.

Figure 1 summarizes the number of *hop-bits* (i.e., *bits* multiplied by the number of hops these bits travel) required in each stage of GDH, where n is the number of participants, b_{GDH} is the size of each intermediate value, H is the number of hops when operational area (A) is calculated as a circle based on a radius (r) with $A = r^2\pi$. As shown in Figure 1, stages 1 and 3 are performed using *unicast*, while stages 2 and 4 employ *broadcast*. We apply different number of hops for *unicast* and *broadcast* in each stage. In stage 1, we assume each node can be reached within one hop so that it can pass a message to the next node in only one hop. In stages 2 and 4, a message is broadcast to all group members, thus taking the average hop distance separating any two nodes into consideration. In stage 3, for simplicity we assume that all members except the sender (M_{n-1}) are located near the boundary of the operational area and the sender broadcasts the message at the center of the operational area. The calculation of the time taken to

perform a rekeying operation due to a join/leave/eviction event based on GDH will be explained later in Section 4.1.

3. System Model

3.1 Assumptions

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, such as GDH, as in MANETs 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 are exponentially distributed with their rates being λ and μ, respectively. The inter-arrival time of data packets issued by a node for group communication is also assumed to be exponentially distributed with rate λ_q. 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 event) or a forced eviction is measured based on GDH [25, 26] to realize distributed key management in MANETs.

We 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 behaviors, considering the possibility of collusion of compromised nodes. Later in Section 4.1, we will explain how to parameterize the linear time attacker function. 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 behavior model above, we use a *linear periodic detection* function to model IDS detection activities which increase linearly with the number of compromised nodes detected. Later in Section 4.1 we will also parameterize the linear periodic detection function.

We assume that *view synchrony* is guaranteed [20] 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. We assume that our GCS enters a security failure state when one of the two conditions stated below is true:

- **Condition C1**: a compromised member, either detected or not, requests and subsequently obtains data using the group key. The system is in a failure state because data have been leaked out to a compromised node, leading the *loss of system integrity* [16] in a security sense.
- **Condition C2**: more than 1/3 of member nodes are compromised by IDS. We assume the Byzantine failure model [11] such that when more than 1/3 of member nodes are compromised, the system fails because of *loss of availability* [16] of system service.

We note that Condition C1 reflects false negatives. On the other hand, Condition C2 reflects false positives. That is, when good nodes are falsely identified as bad nodes and become evicted, the total node population reduces, so is the ratio of good nodes vs. bad nodes. Consequently, it increases the possibility of C2 being satisfied, thereby causing a security failure.

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 available in the system to repair a compromised member and make it a trusted member node again. Initially, all nodes are assumed trusted.

3.2 Attack Model

Host-based IDS and *voting-based IDS* are designed to deal with insider attacks. Outsider attacks (e.g., disrupting traffic, modifying data, eavesdropping, etc.) are dealt with by group key encryption and PKI-based authentication. Insider attacks are due to compromised nodes disguised as legitimate

members to disrupt the system. The following insider attack scenarios are considered following the attack model discussed in [14]:

- An adversary can snoop on the wireless channel to learn of secret information. For example, the adversary can eavesdrop messages sent by vote-participants against a target node, and can disseminate the fake vote result against the target node to all group members.
- An adversary can collude with other compromised nodes so as to more efficiently compromise another node. For example, an adversary can cast a negative vote against a healthy node or cast a positive vote for a compromised node.
- An adversary can attempt to obtain secret information by communicating with other group members with its legitimate group key. When this happens, security failure condition C1 has occurred.
- An adversary can leak the legitimately authorized secret information to outside attackers. Further, an adversary can share their information with other nodes including both outside attackers and inside attackers to more easily compromise other nodes.

3.3 Security Model

Our secure GCS in MANETs meets four requirements in the presence of insider and outsider attacks: *confidentiality*, *integrity*, *availability*, and *authentication*.

Confidentiality is achieved by preserving secrecy properties for secure GCSs. *Group key secrecy* is guaranteed since it is computationally infeasible for an adversary to discover the group key without knowing all intermediate values used in GDH. While *backward secrecy* is preserved, *forward secrecy* is somewhat relaxed for performance gain based on a tradeoff between security and performance requirements. Further, *key independence* is guaranteed since a group key is generated using GDH. In [26], these secrecy properties of GDH have already proven.

For ***integrity***, we use *MAC* (Message Authentication Code) when a message is disseminated. For example, in group communications between members, a MAC (K_G, message) is used using the group key K_G as a secret key. In voting-based IDS, each vote from a vote-participant is disseminated with a MAC, e.g., MAC (K_G, V) where V refers to a vote. Thus, it is impossible for an outside attacker to modify the message without knowing the secret key, K_G, which is only possessed by legitimate members.

Availability is maximized in our scheme by introducing adaptive IDS that dynamically adjusts its intrusion detection interval based on the number of intrusions that have been detected by IDS so as to maximize *MTTSF* of the system.

For ***authenticity***, each member has a private key and its certified public key is available for authentication purposes. When a new member joins a group, the new member's identity is authenticated based on the member public/private key pair by applying the challenge/response mechanism. When a group key is generated through GDH, the source authentication of a participating member is achieved by using the private/public key pair to prevent man-in-the-middle attacks. Moreover, voting-based IDS also uses preloaded public/private key pairs for source authenticity when a vote of each node is disseminated to all group members.

3.4 Metrics

We use *Mean Time to Security Failure* (*MTTSF*) to measure security and *Service Response Time* (*R*) to measure performance properties of our GCS in MANETs as follows:

- ***Mean Time to Security Failure (MTTSF)***: This metric indicates the lifetime of the GCS before it experiences a security failure. For a secure GCS, a security failure occurs when either C1 or C2 defined above is true. As a security metric, a lower *MTTSF* means a faster *loss of system integrity* or *loss of availability*. Therefore, a design goal is to maximize *MTTSF*. We note that the distribution of security failure, and the probability of security breach are also proper security metrics to measure security failure.
- ***Service Response Time (R)*:** This metric refers to the average service response time per group communication operation, including the wireless channel contention delay and transmission delay when a group communication packet is transmitted. This metric is affected by the traffic intensity of rekeying, join/leave/eviction, and IDS operations. A design goal is to find optimal settings to satisfy the system response time requirement *R* while maximizing *MTTSF*.

4. Performance Model

4.1 Stochastic Petri Net Model

We develop a mathematical model based on SPN as shown in Figure 2 to describe the behaviors 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*. Table 1 summarizes the model parameters used.

Table 1: Model Parameters.

Symbol	Meaning
A	Operational area $A = \pi r^2$ (unit: m^2)
R	Radius of an operational area (m)
H	Average number of hops between a sender and a receiver
λ	Arrival rate of join requests (sec^{-1})
μ	Arrival rate of leave requests (sec^{-1})
T_{IDS}	Initial intrusion detection interval (sec)
λ_c	Initial attacker rate (sec^{-1})
m_d	Degree of compromised nodes that have been detected by IDS
$D(m_d)$	A linear detection function that dynamically returns a periodic detection rate based on m_d, i.e., $D(m_d) = m_d(1/T_{IDS})$ (unit: sec^{-1})
m_c	Degree of compromised nodes currently in the system
$A(m_c)$	A linear attacker function based on m_c that dynamically returns the rate at which nodes are compromised, i.e., $A(m_c) = m_c\lambda$ (unit: sec^{-1})
$H_{nb}(m)$	A function that returns the hop number of neighboring nodes based on m
λ_q	Group data communication rate per node (sec^{-1})
$p1$	False negative probability of host-based IDS
$p2$	False positive probability of host-based IDS
T_{cm}	Communication time for broadcasting a rekey message (sec)
b_{GDH}	Length of an intermediate value in applying GDH ($bits$)
b_{GC}	Packet size for group communication activities ($bits$)
m	Number of vote-participants against a target node
BW	Wireless network bandwidth ($Mbps$)
N_{init}	Initial number of member nodes in the system
N	Number of current trusted member nodes
$MTTSF$	Mean time to security failure (sec)
R	Average service response time per group communication operation (sec)
Λ_J	Aggregate group join rate (sec^{-1})
Λ_L	Aggregate group leave rate (sec^{-1})
T_{RTS}	Transmission delay for RTS (request-to-send) (sec)
T_{CTS}	Transmission delay for CTS (clear-to-send) (sec)
$SIFS$	Short inter-frame space (sec)
$DIFS$	Distributed inter-frame space (sec)
T_{slot}	Slot time in random backoff (sec)
$E[CW]$	Average contention-window size (unit: slot)
T_{com}	Transmission delay for a packet (sec)
T_b	Wireless network delay including channel contention time (sec)
T_c	Channel contention delay with an idle channel (sec)
T_{off}	Contention delay due to random backoff when the channel is busy (sec)
Q	Success packet transmission probability without collision occurred
λ_{packet}	Packet arrival rate (sec^{-1})

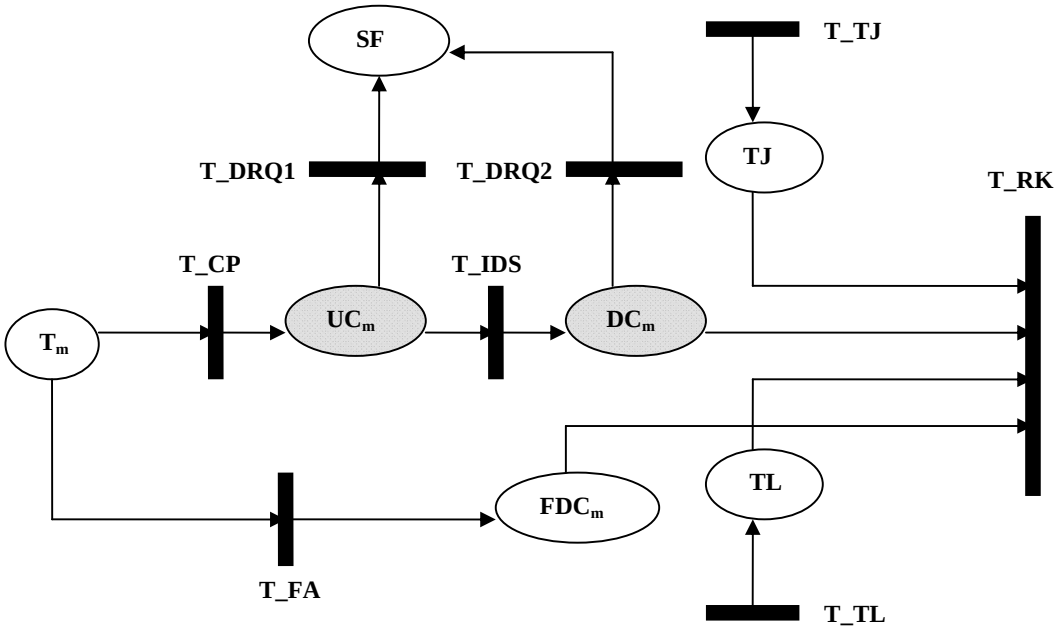

Figure 2: SPN Model.

The SPN model is constructed as follows:

- 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, *TL* holds nodes that have issued a leave request and *SF* represents a system failure state.
- 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. A token in place *SF*, however, does not represent a node of any type, but just represents a system failure state.
- 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. As another example, transitions *T_TJ* and *T_TL* represent join and leave events, respectively, with their rates depending on the population in places T_m and UC_m, that is, $mark(T_m) + mark(UC_m)$, where *mark(X)* returns the number of tokens held in place *X*.

- 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.
- We move nodes (tokens) from one place to another place when an event occurs. For example, after *T_RK* fires, all pending join/leave/eviction operations will be processed by the system. This is modeled by flushing tokens in places FDC_m, DC_m, *TJ*, and *TL*. This is achieved by specifying the "multiplicity" associated with an arc. For example, to evict all nodes in DC_m, the multiplicity of the arc connecting place DC_m and transition *T_RK* is $mark(DC_m)$, so after *T_RK* fires all the tokens (nodes) in place DC_m are flushed, representing that $mark(DC_m)$ nodes have been evicted after a rekeying operation is done. Simultaneously, all tokens (nodes) in other places FDC_m, *TJ*, and *TL* are removed as well.
- 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 modeled 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.
- 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 modeled 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, where *mark(SF)* returns the number of tokens contained in place *SF*.

- 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 modeled 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 $mark(T_m)*D(m_d)*P_{fp}$. Here we 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} in Section 4.1.
- 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 modeled by firing transition *T_RK* for evicting detected compromised members. The rate at which transition *T_RK* fires (for performing a rekeying operation based on GDH) 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.
- The GCS is characterized by member join and leave events, with rates of λ and μ, respectively. This is modeled by associating transitions *T_TJ*, and *T_TL* with these two rates.
- 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 modeled 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 as indicated in Equation (1) below, where *mark* (UC_m) returns the number of compromised but undetected nodes in the system, *mark*(DC_m) returns the number of compromised and detected nodes in the system, *mark*(FDC_m) returns the number of nodes falsely detected as compromised in the system, and *mark*(T_m) returns the number of trusted healthy nodes in the system.

$$\frac{mark(UC_m) + mark(DC_m)}{mark(T_m) + mark(UC_m) + mark(FDC_m) + mark(DC_m)} > \frac{1}{3} \quad (1)$$

4.2 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.

- ***N*:** This is the number of current active group members in the system. 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 $\mu/(\lambda+\mu)$. Let n be the total group population at any time ($n=N_{init}$ at t=0). Then, $N = n\,\lambda/(\lambda+\mu)$. In the SPN model, we initially place $N_{init}\lambda/(\lambda+\mu)$ tokens in place T_m. As the system evolves, N is obtained with *mark* (T_m) + *mark* (UC_m) indicating the number of current active group members.
- **Λ_J & Λ_L:** These are the aggregate join and leave rates of group nodes, respectively. They are also the transitions rates associated with *T_TJ* and *T_TL*. The aggregate leave rate Λ_L is equal to the number of active group members (N) multiplied by per-node join rate (μ). It is easy to see that this aggregate leave rate Λ_L by active members is the same as the aggregate join rate Λ_J by non-active group members.
- **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 follows:

$$if(N > 1) \quad (2)$$

$$T_{cm} = \frac{Nb_{GDH}(2H+1) - b_{GDH}(H+2)}{BW}$$

$$else$$

$$T_{cm} = \frac{b_{GDH}}{BW}$$

Here N again is the number of current member nodes, 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.

- ***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 to transition *T_CP* in our SPN model. Among the three different attacker functions proposed in [8], we adopt the *linear time attacker* function in this paper as follows:

$$A_{linear}(m_c) = \lambda_c \times m_c$$
$$where\ m_c = \frac{mark(T_m) + mark(UC_m)}{mark(T_m)} \quad (3)$$

Here 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.

- ***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 rate to transition *T_IDS* in our SPN model. We parameterize it based on *linear periodic detection* as follows:

$$D_{linear}(m_d) = \frac{1}{T_{IDS}} \times m_d \quad (4)$$
$$where\ m_d = \frac{N_{init}}{mark(T_m) + mark(UC_m)}$$

Here T_{IDS} is a base intrusion detection interval and m_d represents the "degree" of nodes that have been detected by IDS, defined by the ratio of N_{init} to N.

- ***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 -5- gives the expressions for computing P_{fn} and P_{fp} as follows:

$$P_{fp}\ or\ P_{fn} = \sum_{i=0}^{m-\lceil \frac{m}{2} \rceil} \left[\frac{C\binom{mark(UC_m)}{\lceil \frac{m}{2} \rceil + i} \times C\binom{mark(T_m)}{m - \left(\lceil \frac{m}{2} \rceil + i\right)}}{C\binom{mark(T_m) + mark(UC_m)}{m}} \right] + \sum_{i=0}^{m-\lceil \frac{m}{2} \rceil} \left[\frac{C\binom{mark(UC_m)}{i} \times \sum_{j=\lceil \frac{m}{2} \rceil - i}^{m-i} \left[C\binom{mark(T_m)}{j} \times p^j \times C\binom{mark(T_m) - j}{m-i-j} \times (1-p)^{(m-i-j)} \right]}{C\binom{mark(T_m) + mark(UC_m)}{m}} \right] \quad (5)$$

In Equation 5, 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 5 is *p1* when calculating P_{fn} and is *p2* when calculating P_{fp}.

4.3 Assessment of Performance Metrics

MTTSF can be obtained by using the concept of *mean time to absorption* (*MTTA*) in the SPN model. Specifically, we use a reward assignment such that a reward of 1 is assigned to all states except absorbing states which is modeled based on the two security failure conditions (i.e., if either C1 or C2 is met, the system fails). Then the *MTTA* or the *MTTSF* of the system is simply the expected accumulated reward until absorption, $E[Y(\infty)]$, defined as:

$$E[Y(\infty)] = \sum_{i \in S}^{\infty} r_i \int_0^{\infty} P_i(t)dt \quad (6)$$

Here S denotes the set of all states except the absorbing states, r_i (reward) is 1 for those states, and $P_i(t)$ is the probability of state i at time t.

The service response time per group communication packet over the system's lifetime, ***R***, may be calculated by accumulating wireless network delay T_b(t) and transmission delay T_{com}(t) over *MTTSF* divided by *MTTSF*, i.e.,

$$R = \frac{\int_0^{MTTSF} [T_b(t) + T_{com}(t)]dt}{MTTSF} \tag{7}$$

where

$$T_b = T_c + \left(T_c + T_{off}\right) \times (1/Q - 1)$$

$$T_c = T_{RTS} + SIFS + T_{CTS} + SIFS + DIFS$$

$$T_{off} = E[CW] \times T_{slot}$$

$$Q = e^{-\lambda_{packet} \times T_c}$$

$$T_{com} = \frac{(b_{GC} + b_{ack})}{BW}$$

Here T_{com} accounts for the transmission delay for a group communication packet being delivered to the destination, including the time to get an acknowledgement back; b_{GC} is the packet size (*bits*) of a group communication operation and b_{ack} is the packet size (*bits*) for an acknowledgement. T_b accounts for the wireless channel contention time estimated based on *RTS* (request-to-send)/*CTS* (clear-to-send) mechanisms in IEEE 802.11 with *DCF* (distributed coordination function). The contention time depends on the number of retries for securing the wireless channel. Each trial has a basic delay of T_c including the transmission time of the *RTS* and *CTS* packets plus the artificial delay (*SIFS* and *DIFS*) intrinsic to IEEE 802.11. If a trial is not successful, there is a backoff time T_{off} before the next trial is taken place.

While in practice the backoff window size is randomly determined over a range, to simplify our analysis we assume the average window size, denoted by *E [CW]*, is being used in each trial. An attempt is successful if there is no other packet being transmitted during the *RTS/CTS* sequence. Since the overall packet rate is λ_{packet}, assuming packets arrive in accordance with a Poisson process, the probability of no packet arrival during T_c, or the probability of no collision, is given by exp(-$\lambda_{packet}T_c$). By modeling the channel contention process as a geometric distribution with success probability *Q*, the average number of tries before a successful transmission without collision is obtained is given by *1/Q*. We ignore the very small propagation delay in calculating T_b.

5. 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.

Table 2: Parameters and Default Values.

Parameter	Value	Parameter	Value
λ	1/(60*60 s)	m	5
μ	1/(60*60*4 s)	BW	1*Mbps*
T_{IDS}	30 – 9600 (*s*)	N_{init}	60
T_{status}	2 (*s*)	$D(m_d)$	Linear to m_d
λ_c	1/(60*60*12 s)	$A(m_c)$	Linear to m_c
λ_q	1/(60*3 s)	T_{RTS}	0.0003 (*s*)
$p1$	1 %	T_{CTS}	0.0004 (*s*)
$p2$	1 %	$SIFS$	0.00002 (*s*)
b_{GDH}	64 *bits*	$DIFS$	0.00005 (*s*)
b_{GC}	800 *bits*	T_{slot}	0.00005 (*s*)
b_{ack}	32 *bits*	$E[CW]$	256

Table 2 summarizes default parameter values for the base reference system in which the false negative probability (*p1*) and the false positive probability (*p2*) of host-based IDS are set to 1% each since in general less than 1% of false positive or false negative rate is deemed acceptance, reflecting the presence of a medium to high quality host-based IDS. The group communication rate (λ_q) is set to once per 3 minutes. The base compromising rate at which nodes are compromised (λ_c) is once per 12 hours, reflecting a medium-high level of attack strength by the attackers. Later we will vary the values of these key parameters to analyze their effects and sensitivity on system performance. The wireless bandwidth (*BW*) is considered limited and is set at 1Mbps. The ratio of join to leave events (λ:μ) is set to 4, reflecting the fact that nodes join a group much faster than they leave a group. The values used for b_{GDH}, b_{GC} and b_{ack} are set to reflect the number of information bits used for GDH execution, group communication and acknowledgement, respectively. The values used for T_{RTS}, T_{CTS}, *SIFS*, *DIFS*, and T_{slot} are based on *DSSS* for IEEE 802.11 as reported in [1, 4]. The number of vote participants (*m*) in

voting-based IDS is set to 5 for high survivability. Lastly, P_{fn} and P_{fp} of voting-based IDS while not being listed here are to be calculated based on Equation 5.

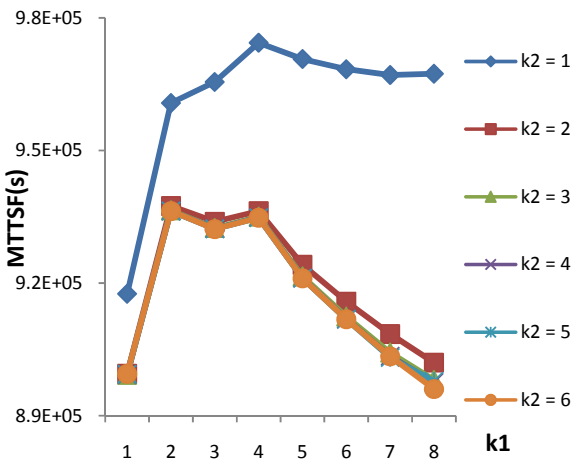

Figure 3: Optimal *k1* and *k2* for *TAUDT* in *MTTSF*.

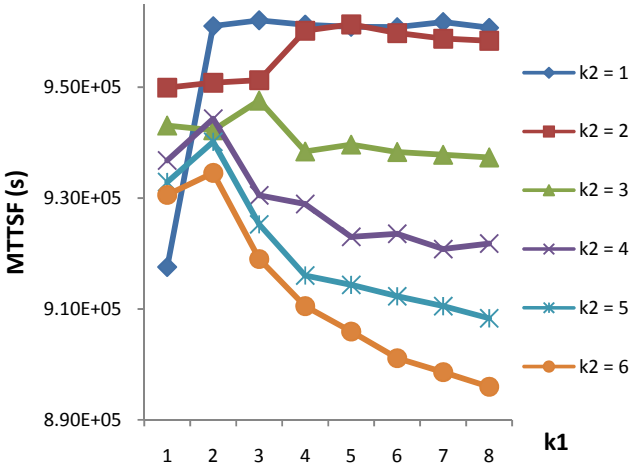

Figure 4: Optimal *k1* and *k2* for *JALDT* in *MTTSF*.

5.1 Optimal Double Thresholds (*k1* and *k2*)

Figures 3 and 4 show the effect of varying *k1* and *k2* on *MTTSF* for *TAUDT* and *JALDT*, respectively. The optimal *MTTSF* in *TAUDT* is observed at (*k1*, *k2*) = (4, 1), as shown in Figure 3. 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 relatively high compared to when *k1* > 1. 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.

The optimal *MTTSF* in *JALDT* is observed at (*k1*, *k2*) = (5, 2), as shown in Figure 4. Recall that in *JALDT k2* governs the threshold for both trusted leave and untrusted leave requests, while in *TAUDT k2* only governs untrusted leave requests. Consequently, the optimal *k2* is at 2 in *JALDT* as opposed to the optimal *k2* at 1 in *TAUDT*. The reason that the optimal *k1* is at 5 in *JALDT* is that *k1*=5 (as opposed to 4) best balances the probability of security failure due to Condition 1 vs. Condition 2, as explained earlier, since *k1* now only governs join operations.

Figures 5 and 6 show the effect of *k1* and *k2* on the service response time, *R*. The trends shown in Figures 5 and 6 strikingly reflect the overall communication cost per time unit (*s*) vs. *k1* and *k2* (not shown here for brevity). In Figure 5, we see the optimal (*k1, k2*) is at (4, 1) being identical to that in Figure 3. In Figure 6, we also observe that the optimal (*k1, k2*) is at (5, 2) being identical to that in Figure 4. The existence of the optimal (*k1, k2*) setting can be explained in a similar way as we have done for Figures 3 and 4.

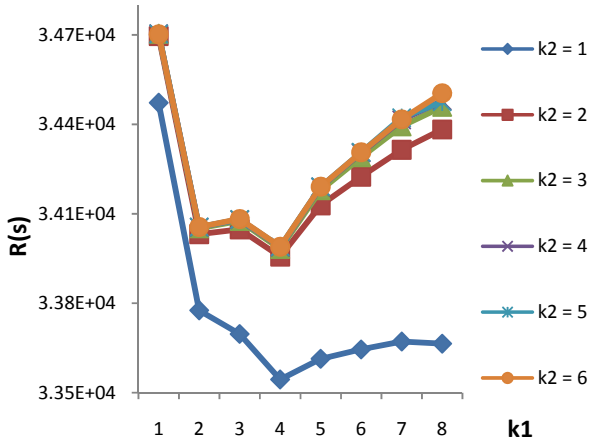

Figure 5: Optimal *k1* and *k2* for *TAUDT* in Service Response Time *R*.

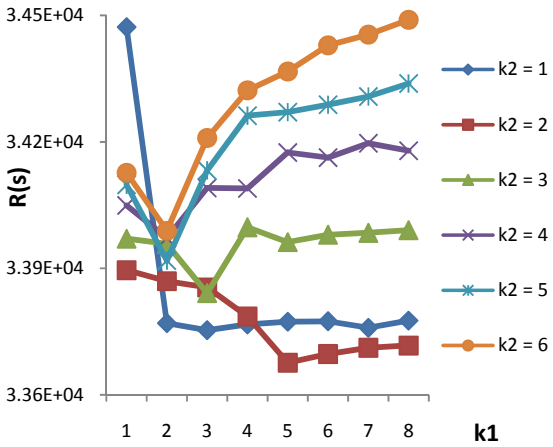

Figure 6: Optimal *k1* and *k2* for *JALDT* in Service Response Time *R*.

5.2 Optimal Intrusion Detection Intervals (T_{IDS})

Here we analyze optimal intrusion detection intervals (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 7 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 is increased 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 relatively high compared to the other two rekeying protocols. 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 7.

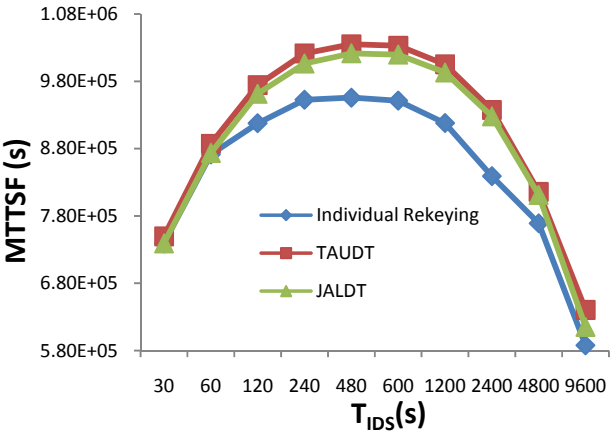

Figure 7: Optimal T_{IDS} in *MTTSF*.

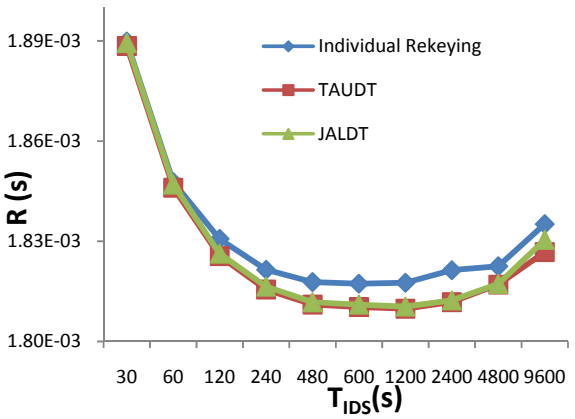

Figure 8: Optimal T_{IDS} in *R*.

Figure 8 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 reason that *R* goes up as T_{IDS} increases past the optimal point is that a larger T_{IDS} leads to more activities

during the IDS period because more bad nodes would be remaining in the system without being detected by IDS. These bad nodes engage in group communication, status exchange, and voting activities as good nodes, thereby causing a higher contention of the wireless channel and a higher service response time. On the other hand, when T_{IDS} is very small, the communication overhead due to IDS dominates and consequently R is also high. We note that, however, the variation in R is small overall and is relatively insensitive to the intrusion detection interval. Among the three curves in Figure 8, we again 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.

5.3 Sensitivity Analysis

In this section, we perform sensitivity analysis to test the sensitivity of *MTTSF* and *R* vs. T_{IDS} with respect to certain key parameters including λ_c, λ_q, and (*p1, p2*). We use *TAUDT* under optimal (*k1, k2*) as the base case since it has been identified it as the best scheme in Section 5.2.

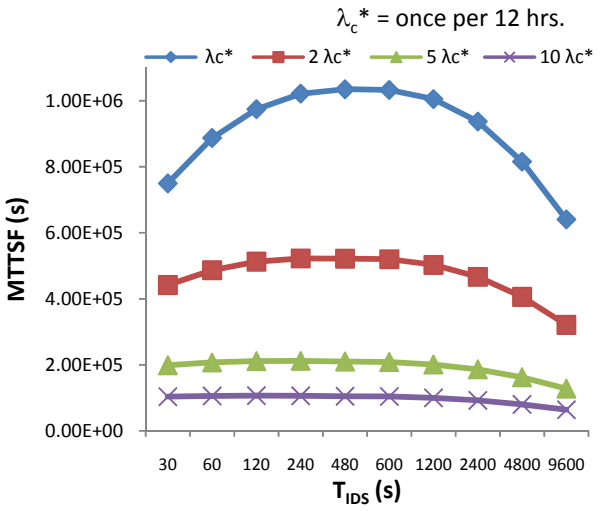

Figure 9: Sensitivity of *MTTSF* vs. T_{IDS} with respect to λ_c.

Figure 9 shows the sensitivity of *MTTSF* vs. T_{IDS} with respect to the compromising rate (λ_c) which varies from λ_c* to 10λ_c* covering an order of magnitude change. We observe that as λ_c increases, *MTTSF* decreases because a higher λ_c will cause more compromised nodes to be present in the system.

Consequently, the optimal T_{IDS} value that maximizes *MTTSF* decreases because more compromised nodes will exist as λ_c increases and the system will need to execute IDS more frequently to maximize *MTTSF*. Nevertheless, we observe that the optimal T_{IDS} value that maximizes *MTTSF* is sensitive to λ_c only when the order of magnitude of λ_c changes (e.g., when its value changes from λ_c* to $10\lambda_c$*) but is relatively insensitive to λ_c when its order of magnitude remains the same (e.g., when its value changes from λ_c* to $2\lambda_c$*). We attribute this level of sensitivity to the way our detection function (see Equation 4) reacts to the attacker strength (see Equation 3) linearly.

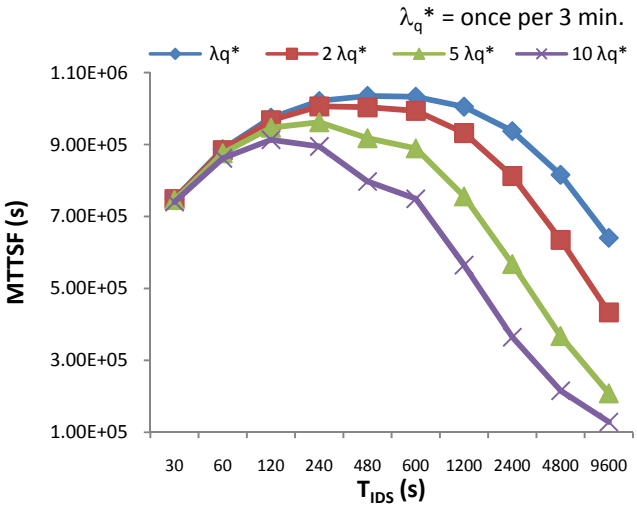

Figure 10: Sensitivity of *MTTSF* vs. T_{IDS} with respect to λ_q

Figure 10 shows the sensitivity of *MTTSF* vs. T_{IDS} with respect to the group communication rate (λ_q). We observe that when λ_q is low so the data-leak attack is not performed often, the positive effect of IDS is pronounced, leading to a high *MTTSF*. On the other hand, when λ_q is high so the data-leak attack is frequent, the negative effect of IDS is pronounced, so *MTTSF* is low. We also observe that the optimal T_{IDS} becomes smaller as λ_q increases because the system prefers removing compromised nodes as soon as possible so that compromised nodes would not have a chance to perform data-leak attacks. Another observation is that when T_{IDS} is sufficiently small, e.g., $T_{IDS} < 120$ *s*, *MTTSF* remains about the same regardless of the magnitude of λ_q. This is because when IDS is being invoked too frequently, the adverse effect of false positives dominates the positive effect of IDS. Lastly we observe that the optimal T_{IDS} value is sensitive to λ_q even when its order of magnitude remains the same (e.g., when λ_q value changes from λ_q* to $2\lambda_q$*). We attribute this level of sensitivity to the way voting-based IDS reacts to data-leak attacks (i.e., to avoid Condition C1 from being satisfied).

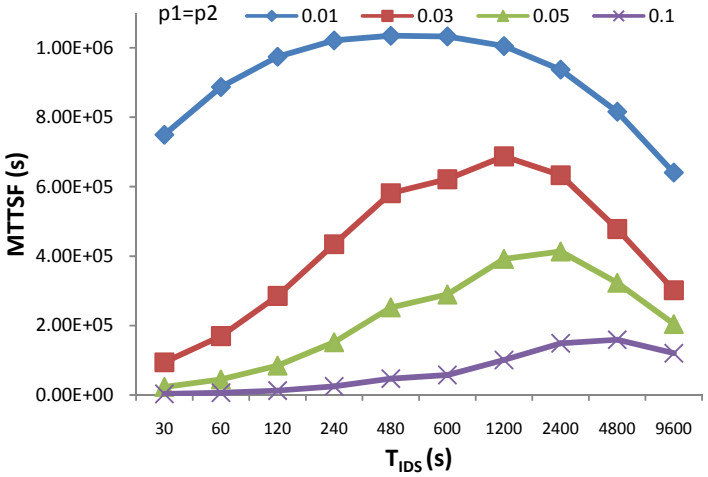

Figure 11: Sensitivity of *MTTSF* vs. T_{IDS} with respect to (*p1*, *p2*).

In Figure 11, we check the sensitivity of *MTTSF* vs. T_{IDS} with respect to host-based IDS false negative and false positive probabilities, i.e., (*p1, p2*). We see that when the IDS quality is low as indicated by high *(p1, p2)* values (e.g., the last curve on Figure 11), *MTTSF* is low, in which case a large T_{IDS} would be preferred because the system can delay generating false positives by the low-quality IDS as much as possible with a long IDS interval. We observe that in general the optimal T_{IDS} value is very sensitive to (*p1, p2*) even when their order of magnitude is the same (e.g., when their values change from 0.01 to 0.03). We attribute this acute sensitivity to the way voting-based IDS reacts to host-based IDS false negative and false positive probabilities by acutely adjusting the detection interval to maximize *MTTSF*.

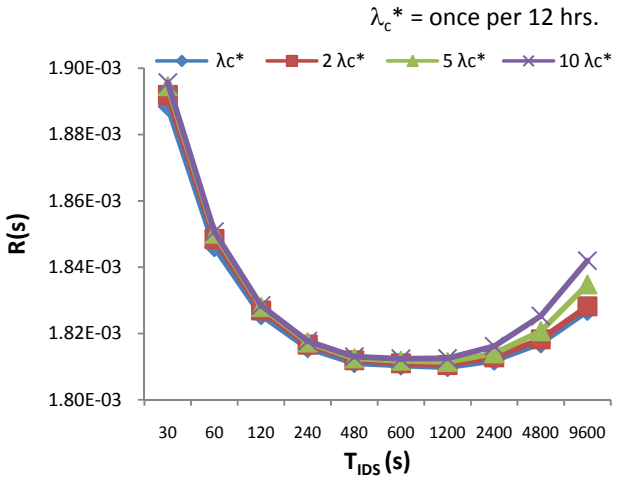

Figure 12: Sensitivity of *R* vs. T_{IDS} with respect to λ_c.

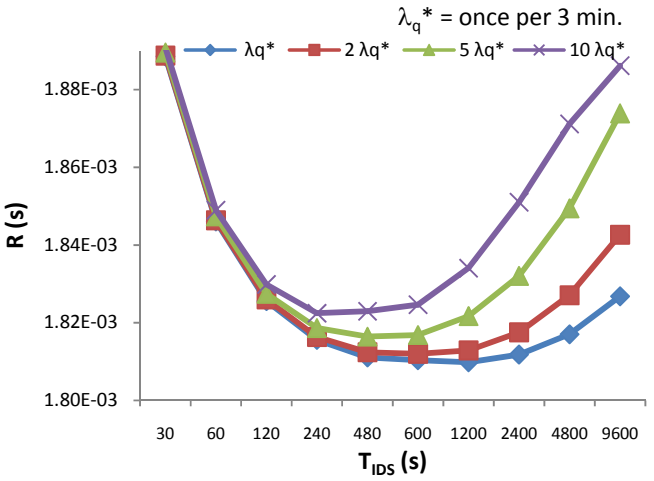

Figure 13: Sensitivity of *R* vs. T_{IDS} with respect to λ_q.

We repeat the same sensitivity analysis to test the effects of λ_c, λ_q, and (*p1, p2*) on *R* vs. $T_{IDS.}$ The results are shown in Figures 12-14 for λ_c, λ_q, and (*p1, p2*), respectively. In Figure 12, we observe that as λ_c increases, *R* increases due to more compromised nodes being evicted and thus there is more traffic being generated for rekeying. However, the optimal T_{IDS} that minimizes *R* is relatively insensitive to λ_c because the traffic generated for rekeying does not dominate other sources of traffic in the system. In Figure 13, we observe that as λ_q increases, *R* increases due to a higher level of group communication activities. To minimize *R* in the presence of a high λ_q value, the system would use a small T_{IDS} so as to more quickly detect and evict truly or falsely identified compromised nodes from the system to reduce the total population and the net traffic. We observe that the optimal T_{IDS} that minimizes *R* is sensitive to λ_q values in the same order of magnitude because the system must acutely balance the extra traffic introduced due to more frequent IDS and eviction activities (as a result of the use of a smaller T_{IDS}) vs. the traffic being reduced due to less group communication and status exchange activities (as a result of the decreasing population because of fast eviction). In Figure 14, we first observe that as (*p1, p2*) values increase, *R* decreases. This is because low-quality IDS characterized by high (*p1, p2*) values will likely evict compromised nodes (albeit mostly falsely-identified) faster than high-quality IDS characterized by low (*p1, p2*) values. As a result, the node population and group communication traffic in the system will be greatly reduced. Consequently, to minimize *R* in the presence of high (*p1, p2*) values, the system would use a small T_{IDS} to further accelerate the reduction of the total population and the net traffic. Here we observe that the optimal T_{IDS} that minimizes *R* is sensitive to (*p1, p2*) values in the same order of magnitude. We again attribute this level of sensitivity to the system's ability to acutely determine the optimal T_{IDS} that can best balance the traffic sources as (*p1, p2*) varies.

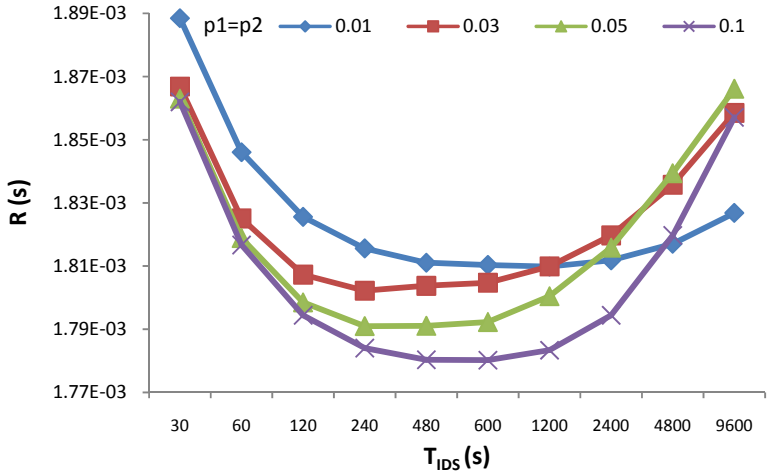

Figure 14: Sensitivity of *R* vs. T_{IDS} with respect to (*p1, p2)*.

6. Applicability

To apply the analysis results obtained in the paper, one can summarize findings into a table listing optimal batch rekeying and intrusion detection intervals covering a range of parameter values characterizing perceivable operational and environmental conditions. Then, at runtime, the system can perform a table lookup operation to select the best batch rekey and intrusion detection intervals based on statistical information collected dynamically.

While we have exemplified with batch rekeying and host-based/voting-based IDS as the rekeying and IDS algorithms in this paper, the mathematical model developed is generally applicable to other types of rekeying and IDS algorithms. The changes can be reflected by means of parameterization (giving proper model parameter values). For example, if we consider a network environment in which a centralized key server and network-based IDS are employed, we can simply replace P_{fp} and P_{fn} with *p1* and *p2*. If we consider other rekeying algorithms, centralized or decentralized, or distributed key management protocols, all one has to do is to redefine the rekeying conditions based on the state information provided in the SPN model, e.g., based on the number of join/leave/eviction operations in a state. The performance metric calculation and methodology developed remain same for identifying optimal design conditions that maximize *MTTSF*.

7. Conclusion and Future Work

In this paper, we investigated the design of integrating intrusion detection with batch rekeying to cope with both outsider and insider attacks for GCSs in MANETs, and analyzed the tradeoff between security and performance properties of the resulting GCS due to the use of these two protocols. We showed that there exist optimal settings in terms of batch rekey intervals (*k1* and *k2*) and intrusion detection intervals under which the system lifetime (in terms of *MTTSF*) is maximized while performance requirements (in terms of service response time) is satisfied.

The current work considers the case in which the node density is high, and thus all nodes are in one group and will not be partitioned in MANETs. In the future, we plan to extend this work to consider the case in which a GCS may be partitioned due to mobility or changes of transmission range because of energy depletion. We also plan to integrate IDS and batch rekeying with hierarchical key management [7] for achieving high scalability, configurability and survivability for GCSs in MANETs.

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. of Symp. on Applications and the Internet Workshops*, 27-31 Jan. 2003, pp. 368 – 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] M. M. Carvalho, J. J. Garcia-Luna-Aceves, "Delay Analysis of IEEE 802.11 in Single-hop Networks," *Proc. of 11th IEEE Int'l Conf. on Network Protocols,* 4-7 Nov. 2003, pp. 146-155.

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

[6] J. H. Cho, I. R. Chen, and M. Eltoweissy, "On Optimal Batch Rekeying for Secure Group Communications in Wireless Networks," *Wireless Networks (ACM/Springer)*, vol. 14, no. 6, Dec. 2008, pp. 915-927.

[7] J. H. Cho, I. R. Chen, and D.C. Wang "Performance Optimization of Region-Based Group Key Management in Mobile Ad Hoc Networks," *Performance Evaluation (Elsevier),* vol. 65, no. 5, May 2008, pp. 319-344.

[8] 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 (NPSec),* Boston, Nov. 2005, pp. 13-18.

[9] J. H. Cho, I.R. Chen, and P.G. Feng, "Performance Analysis of Dynamic Group Communication Systems with Intrusion Detection Integrated with Batch Rekeying in Mobile Ad Hoc Networks," *Proc. of 22nd Int'l Conf. on Advanced Information Networking and Applications-Workshop(AINAW2008),* GinoWan, Okinawa, Japan, 25-28 Mar. 2008, pp. 644 - 649.

[10] H. Debar and A. Wespi, "Aggregation and Correlation of Intrusion-Detection Alerts," *Proc. of 4th Int'l Symp. Recent Advances in Intrusion Detection*, 10-12 Oct. 2001, pp. 85-103.

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

[12] T. Hardjono, B. Cain, and I. Monga, *Intra-Domain Group Key Management Protocol*, Internet Draft, Feb. 1998.

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

[14] K. Inkinen, "New Secure Routing in Ad Hoc Networks: Study and Evaluation of Proposed Schemes," *Seminar on Internetworking*, Sjökulla, Finland, 2004.

[15] O. Kachirski and R. Guha, "Intrusion Detection using Mobile Agents in Wireless Ad Hoc Networks," *Proc. of IEEE Workshop on Knowledge Media Networking*, 10-12 July 2002, pp. 153-158.

[16] T. Karygiannis and L. Owens, "Wireless Network Security: 802.11, Bluetooth and Handheld Devices," NIST, 2002, http://www.windowsecurity.com/articles/intrusion_detection/.

[17] P. Kazienko and P. Dorosz, Intrusion Detection Systems (IDS) Part I, II: Network Intrusions, Attack Symptoms, IDS Tasks, and IDS Architecture, 2004.

[18] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad Hoc Networks," *IEEE 9th Int'l Conf. on Network Protocols (ICNP'01)*, 11-14 Nov. 2001, pp. 251-260.

[19] R. Lang and Z. Deng, "Data Distribution Algorithm Using Time Based Weighted Distributed Hash Tables," *Proc. of 7th Int'l Conf. on Grid and Cooperative Computing,* 24-26 Oct. 2008, pp. 210-213.

[20] P. P. C. Lee, J. C. S. Lui, and D. K. Y. Yau, "Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups," *IEEE/ACM Trans. on Networking*, vol. 14, no. 2, Apr. 2006, pp. 263-276.

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

[22] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *Proc. of 6th Annual ACM/IEEE Mobile Computing and Networking*, Boston, MA, Aug. 2000, pp.255-265.

[23] A. Mishra, K. Nadkarni, and A. Patcha, "Intrusion Detection in Wireless Ad-hoc Networks," *IEEE Wireless Communications,* vol. 11, no. 1, Feb. 2004, pp.48-60.

[24] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A Scalable Group Rekeying Approach for Secure Multicast," *IEEE Symp. on Security and Privacy*, Oakland, CA, May 2000, pp. 215-228.

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

[26] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," *Proc. of 3rd ACM Conf. on Computer and Communications Security*, New Delhi, India, Jan. 1996, pp. 31-37.

[27] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C. Y. Tseng, T. Bowen, "A General Cooperative Intrusion Detection Architecture for MANETs," *Proc. of 3rd IEEE Int'l Workshop on Information Assurance*, 23-24 Mar. 2005, pp. 57-70.

[28] B. Sun, K. Wu, and U.W. Pooch, "Alert Aggregation in Mobile Ad Hoc Networks," *Proc. of 2003 ACM Workshop on Wireless Security*, ACM Press, San Diego, Sep. 2003, pp. 69-78.

[29] B. Sun, K. Wu, and U.W. Pooch, "Routing Anomaly Detection in Mobile Ad-hoc Networks," *Proc. of IEEE 12th Int'l Conf. on Computer Communications and Networks*, ICCCN, Oct. 2003, pp. 25-31.

[30] C.K. Wong, M. Gouda, and S.S. Lam, "Secure Group Communications Using Key Graphs," *IEEE/ACM Trans. on Networking,* vol. 8, no. 1, Feb. 2000, pp. 16-30.

[31] Y.R. Yang, X. Li, X. Zhang, S.S. Lam, "Reliable Group Rekeying: A Performance Analysis," *ACM SIGCOMM 2001*, San Diego, CA, Aug. 2001. pp. 27-38.

[32] H. Zhang, A. Goel, and R. Govindan, "Improving Lookup Latency in Distributed Hash Table Systems using Random Sampling," *IEEE/ACM Trans. on Networking,* vol. 13, no. 5, Oct. 2005, pp. 1121-1134.

[33] Y. Zhang, W. Lee, and Y.A. Huang, "Intrusion Detection Techniques for Mobile Wireless Networks," *Wireless Networks (ACM/Springer),* vol. 9, no. 5, Sep. 2003, pp. 545-556.

[34] B. Zhu, F. Bao, R. H. Deng, M. S. Kankanhalli, and G. Wang, "Efficient and Robust Key Management for Large Mobile Ad-hoc Networks," *Computer Networks*, vol. 48, no. 4, July 2005, pp. 657-682.

[35] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," *7th ACM Conference on Computer and Communications Security*, Athens, Greece, Nov. 2000, pp.235-244.

[36] B. Wu, J. Wu, and Y. Dong, "An efficient group key management scheme for mobile ad hoc networks," *International Journal of Security and Networks*, Vol. 4, No. 1, 2009, pp. 125-134.