GKMPAN: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks

Sencun Zhu[1] Sanjeev Setia[1] Shouhuai Xu[2] Sushil Jajodia[1]

[1]Center for Secure Information Systems, George Mason University, Fairfax, VA 22030
[2]Department of Computer Science, University of Texas at San Antonio, San Antonio, TX 78249
E-mail: {szhu1,setia,jajodia}@gmu.edu,shxu@cs.utsa.edu

Abstract

We present GKMPAN, an efficient and scalable group rekeying protocol for secure multicast in ad hoc networks. Our protocol exploits the property of ad hoc networks that each member of a group is both a host and a router, and distributes the group key to member nodes via a secure hop-by-hop propagation scheme. A probabilistic scheme based on pre-deployed symmetric keys is used for implementing secure channels between members for group key distribution. GKMPAN also includes a novel distributed scheme for efficiently updating the pre-deployed keys. GKMPAN has three attractive properties. First, it is significantly more efficient than group rekeying schemes that were adapted from those proposed for wired networks. Second, GKMPAN has the property of partial statelessness*; that is, a node can decode the current group key even if it has missed a certain number of previous group rekeying operations. This makes it very attractive for ad hoc networks where nodes may lose packets due to transmission link errors or temporary network partitions. Third, in GKMPAN the key server does not need any information about the topology of the ad hoc network or the geographic location of the members of the group. We study the security and performance of GKMPAN through detailed analysis and simulation.*

1. Introduction

Many applications of ad hoc networks involve collaborative computing among a large number of nodes and are thus group-oriented in nature. For deploying such applications in an adversarial environment such as a battlefield or even in many civilian commercial scenarios, it is necessary to provide support for secure group communication. In this paper, we address the issue of providing confidentiality for group communication in ad hoc networks.

The most efficient approach for achieving confidential group communication is to use a symmetric group key that is shared by all the nodes for data encryption. This approach however introduces the problem of *group rekeying*, i.e., the group key must be updated and redistributed to all the remaining nodes in a secure, reliable and timely fashion when group membership changes. This problem has been studied extensively in the context of secure multicast in wired networks and several scalable key management protocols have been proposed, e.g., OFT [2], Subset-Difference [12], LKH [19], and ELK [15]. However, these approaches are not directly applicable to ad hoc networks, because the communication cost per node can become very high for a large ad hoc network with very dynamic group membership. So far very few group rekeying schemes have been proposed for ad hoc networks. They either use public-key techniques [7], or adapt the LKH scheme for ad hoc networks [10]. Public-key based schemes [7] are more expensive than symmetric-key based schemes in both communication and computation. The adapted LKH scheme [10] incurs the computational and communication cost that is of the same order as the LKH scheme [19]. Moreover, the LKH-based schemes [2, 10, 19] have the disadvantage of *stateful* group rekeying schemes that a node that has missed a group rekeying operation will need to obtain previously transmitted key encryption keys in order to be able to decrypt the current group key. This may involve requesting the key server for retransmission of any missing key encryption keys, which is highly undesirable in a multi-hop wireless network.

In this paper, we present a scalable and efficient group rekeying protocol (GKMPAN) for ad hoc networks based solely on symmetric key techniques. GKMPAN exploits the property of an ad-hoc network that member nodes are both hosts and routers. In IP Multicast, all group members are end hosts, and they have no responsibility for forwarding keying materials to other group members. In contrast, for group communication in an ad hoc network, the members of the group also act as routers. As such, in GKMPAN the key server only has to deliver the new group key securely to the group members that are its immediate neighbors, and

Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)
0-7695-2208-4/04 $20.00 © 2004 IEEE

these neighbors then forward the new group key securely to their own neighboring members. In this way, a group key can be propagated to all the members. Because every node only needs to receive one encryption of the group key, the average transmission cost per node is one key independent of the group size.

For the above scheme to work, a fundamental requirement is the existence of a secure channel between every pair of neighboring nodes. GKMPAN provides secure channels through probabilistic key pre-deployment. The technique of probabilistic key pre-deployment has been applied in several studies [3, 4, 23]; however, to the best of our knowledge, none of these studies address the issue of updating the pre-deployed keys. Updating the predeployed keys is critical in order to prevent the compromised and revoked nodes from launching a collusive attack in which they pool together their keys with the goal of jeopardizing the secure channels between other nodes. Without key updating, both the performance and security of the system will degrade greatly with the number of compromised nodes. To address this issue, we present an efficient distributed key updating scheme for updating any compromised channels.

The contributions of this paper are two-fold. First, GKMPAN, which based on symmetric key techniques, is significantly more communication-efficient than previous approaches [10, 19] for group rekeying when used for ad hoc networks. It also has the property of *partial statelessness*, that is, a node can decode the current group key even if it has missed a certain number of previous group rekeying operations. This makes it very attractive for ad hoc networks where nodes may lose packets due to transmission link errors or temporary network partitions. Moreover, unlike the LKH-based scheme in [10], GKMPAN does not require any information about the topology of the ad hoc network or the geographic location of the members of the group.

Second, the key update scheme of GKMPAN can also be used to increase the robustness of other probabilistic key pre-deployment based schemes [3, 4, 23] when compromised nodes can be detected.

The rest of this paper is organized as follows. An overview and the protocol details of GKMPAN are presented in Section 2. We analyze the security of GKMPAN in Section 3 and evaluated its performance in Section 4. The related work are described in Section 5. Finally, the conclusion is made in Section 6.

2. The Group Rekeying Protocol

In this section, we first discuss our assumptions and present a brief overview of our protocol, then describe its operation in greater details.

2.1. Network and Security Assumptions

Network Assumptions For the ease of presentation, in this section, we assume that all nodes in the ad hoc network are members of the group. We discuss how this scheme can be extended for networks where not all nodes are members of a group in Section 2.4.3. The communication model we consider is group-oriented communication; that is, messages are addressed to all the members. Note that using pairwise shared keys for securing group communication does not improve security in comparison to a scheme based on group keys. This is because under both schemes an adversary only needs to compromise one node to obtain the group data; moreover, if pairwise keys are used for securing group data, a node will have to perform decryptions and re-encryptions for the data packets it is forwarding. Nevertheless, if the network needs to provide pairwise keys for private communication between pairs of nodes, we can directly employ the probabilistic pairwise key establishment scheme in [23] without making any additional security and network assumptions.

We assume that the resources of a node, such as power, computational and communication capacity, and storage are relatively constrained; thus a node neither can afford public-key operations nor has space for storing pre-deployed pairwise shared keys for all the nodes in the network. However, we assume that every node has space for storing hundreds of bytes or a few kilobytes of keying materials, depending on the security requirements. One type of such nodes is the current generation of sensor nodes (e.g., Berkeley Mica2 motes [20] with 8MHZ CPU and 4K RAM).

Security Assumptions and Attack Models We assume that there is a group manager (or multiple collaborative managers for robustness) managing the group membership. Under our protocol, group rekeying is initiated by the group manager to revoke one or multiple nodes. We do not specify the cause (e.g., policy change, owner compromises) for node revocation. In particular, we do not assume that the reason for node revocation must be node misbehavior (e.g, injecting spurious packets). Unlike sensor networks where nodes are often unattended, in a mobile ad hoc network, it is more common that nodes are carried by other entities (e.g., soldiers, vehicles). Therefore, a node revocation is often the result of revoking the carrier of a node. For example, if a solider is captured by the adversary or is missing in a battlefield, other soldiers can report the event to the group manager, which initiates a group rekeying operation to revoke the node carried by this solider.

We do not distinguish between an attacker and a compromised node, because we assume that an attacker can obtain all the information stored in a compromised node. We assume, however, that a non-compromised node can be trusted; that is, a node executes the protocol correctly un-

less it is compromised. Finally, we assume that compromised nodes are detected immediately and no new nodes are compromised before the current group rekeying is completed. In the extended version of this paper [22], we discuss a potential colluding attack against this scheme when the above assumptions do not hold, and we also present an extension of this scheme to address the attack.

Since wireless communication is broadcast-based, we assume that an adversary can eavesdrop on all traffic, inject packets, and replay older packets. Since we assume that an adversary can take full control of compromised nodes, an adversary may command compromised nodes to drop or alter messages going through them.

2.2. Design Goals

Given the threat model described above, in this paper, we focus on preventing a *group key recovery attack* in which an attacker's goal is to learn the group key through eavesdropping on key distribution messages exchanged by group members. Several nodes whose group membership has been revoked can collude in this attack by pooling together their keys. Our goal is to design an efficient group rekeying scheme that updates the compromised keys efficiently once the compromised nodes are detected. The scheme should enable the non-compromised nodes to reject spurious group keys injected by compromised nodes. The scheme should also be robust to refusal-of-service attacks in which compromised nodes prevent other nodes from receiving group keys by dropping packets going through them.

2.3. Protocol Overview

Our group rekeying protocol involves a *key pre-distribution* phase and multiple *rekeying* phases.

Key Pre-distribution Prior to the deployment of the ad hoc network, all nodes obtain a distinct subset of keys out of a large key pool from the key server and these keys are used as key encryption keys (KEKs) for delivering group keys.

A rekeying operation itself involves three steps: authenticated revocation notification, secure group key distribution and key updating.

Authenticated Node Revocation When the key server decides to revoke a node, it broadcasts a revocation notice to the network.

Secure Group Key Distribution The key server generates and distributes a new group key K. The key K is propagated to all the remaining nodes in a hop-by-hop fashion, secured with the non-compromised predeployed keys as KEKs.

Key Update After a node receives and verifies the group key K, it updates its own KEKs based on K.

2.4. The Protocol in Detail

Notations Below are the notations that appear in the rest of this discussion.

- u, v (in lower case) are principals such as communicating nodes.
- R_u is a set of keys that u possesses, and I_u is the set of key ids corresponding to the keys in R_u.
- $\{f_k\}$ is a family of pseudo-random functions [5].
- $\{s\}_k$ means encrypting message s with key k.
- $MAC(k, s)$ is the MAC of message s using a symmetric key k.

2.4.1. Key Pre-distribution Each node is loaded with the following information:

1. Each node u is loaded with R_u, which contains m distinct keys chosen from the key pool P of l keys $\{k_1, k_2, \ldots, k_l\}$, and these keys are used as KEKs. A deterministic algorithm is used to decide I_u, the set of ids of the keys in R_u. Specifically, for each node, the group manager generates m distinct integers between 1 and l using a pseudo-random number generator upon the input of a node id. This construction allows any node that knows another node's id u to determine I_u. Each key in the key pool has a probability of m/l to be chosen by each node.
2. Each node is loaded with the initial group key k_g that is used for securing group-wide communications, and an individual key that is only shared between the node and the key server.
3. Finally, each node is loaded with the commitment (i.e., the first key) of the key chain of the key server because we are employing TESLA [14] for broadcast authentication.

Our scheme does not require a key pre-distribution phase for every instance of network formation. Indeed, there is no limit on how many times these pre-distributed keys can be used securely because our rekeying scheme (detailed below) updates these keys securely after every group rekeying. Also note that this key pre-distribution phase can be considered equivalent to the member joining phase in traditional secure IP multicast. We discuss how additional nodes can be added to the network after the initial key pre-distribution phase in Section 2.4.3.

2.4.2. Group Rekeying In this section, we first introduce the main group rekeying operations, followed by the details of the involved procedures. Let u be the node to be revoked.

Hence all the keys in R_u and the group key k_g are considered to be compromised and must be updated securely. The group rekeying involves the following steps.

1. The key server determines M, the id of the non-compromised key that is possessed by the maximum number of remaining nodes in the network. The key server then generates an intermediate key $k_{im} = f_{k_M}(k_g)$ and the new group key $k'_g = f_{k_{im}}(0)$. Then it broadcasts a node revocation message that contains M, the id of the revoked node (i.e., u), and $f_{k'_g}(0)$. The message is authenticated by TESLA (see the *authenticated node revocation* process).
2. The nodes that possess k_M can compute the intermediate key k_{im} independently after they verify the above message. Then they (and also the key server) forward k_{im} to each of their neighbors that do not possess k_M via logical paths (see the *logical path discovery* process) secured by non-compromised keys, i.e., keys in $P - R_u$. Node u will not receive k_{im} even though it can impersonate a non-revoked node v by claiming node v's id, because none of the keys in R_u are used while establishing logical paths.
3. Every node computes the new group key $k'_g = f_{k_{im}}(0)$. It verifies the correctness of k'_g by checking if $f_{k'_g}(0)$ equals to that in the node revocation message. Node u cannot compute k'_g because neither it has k_M nor it receives k_{im} from others.
4. Every node, say v, updates every key k_i in R_v as $k'_i = f_{k_i}(0)$. We denote the updated set of keys, k'_i's, as R'_v.
5. If k'_i is computed from k_i that was held by node u, i.e., $k_i \in R_u$, the nodes that hold k'_i further update k'_i as $k'_i = f_{k_{im}}(k'_i)$.
6. Finally, every node erases k_{im} and the original k_i's.

To understand this scheme, let us consider the *first* revocation in the network when node u is revoked. After step 2, every node except u has k_{im}. In step 3 and step 5, k_{im} is used to update the group key and the compromised keys (i.e., the keys in R_u). Since we use a pseudo-random function to update these keys, knowing the updated keys does not help an attacker compute k_{im}. Moreover, since the distribution of k_{im} in step 2 is secured by the non-compromised keys in P, updating all these non-compromised keys in step 4 and then erasing the original keys in step 6 prevents a node that is compromised in the future from providing any keys that enable the recovery of k_{im}.

Note that in step 6, every node deletes k_{im}; therefore, in the entire network, no node holds k_{im} and no node can recover it when the rekeying process is completed. Thus, all the keys in R_u are updated securely. In fact, the status of the system is reinstated to its original setting after every rekeying. Even if all the previously revoked nodes collude, they cannot compute the updated keys.

The probability that a particular key K is allocated to a node in the key pre-distribution phase is equal to m/l. Therefore, on average, the number of nodes that possess K is given by Nm/l for a group size of N. Recall that k_M is selected to be the key possessed by the maximum number of nodes. Thus, the number of nodes in the network that possess k_M is generally larger than Nm/l. Therefore, a fraction of nodes can compute the new group key independently without waiting for it to be delivered to them. To reduce the rekeying latency and increase the reliability of delivering the new group key, these nodes can also independently start propagating the new group key to their neighbors.

Logical Path Discovery The *logical path discovery* process is necessary when a node wants to forward a new group key to its neighbors securely. We say there are logical paths between two nodes when (i) the two nodes share one or more keys. We call such paths *direct* paths. (ii) the two nodes do not share any keys, but through other nodes they can transmit messages securely to each other. We call such paths *indirect* paths and call the involved nodes *proxies*.

In our design, it is very easy to find logical paths between two nodes. Since the key pre-distribution algorithm is public and deterministic, a node u can independently compute I_v, the set of key ids corresponding to a node v's key set. Therefore, without pro-actively exchanging the set of its key ids with others, a node knowing the ids of its neighbors can determine not only which neighbors share or do not share keys with it, but also which two neighbors share which keys. The latter knowledge is very valuable when node u does not share any keys with a neighbor node v, because node u can ask a neighbor (say x) which shares keys with each of them to act as a *proxy*. For example, suppose node u shares a key k_{ux} with node x, node v shares a key k_{vx} with node x, but no shared key exists between node u and node v. To forward an intermediate key k_{im} to node v, the following steps are taken.

$$u \rightarrow x : \{k_{im}\}_{k_{ux}},\ x \rightarrow u : \{k_{im}\}_{k_{xv}},\ u \rightarrow v : \{k_{im}\}_{k_{xv}}$$

From this example, we can see that a *proxy* node acts as a translator between nodes. We call node x in the above example node u's *one-hop proxy* to v. More generally, node x is said to be node u's *i-hop proxy* if x is i hops away from u and x shares a key with both u and v. If u and v do not have any *direct* paths and indirect paths via a one-hop proxy to each other, they can resort to using a *multi-hop proxy*. GKMPAN always uses any direct paths that exist between nodes in preference to indirect paths, since the use of an indirect path incurs additional computational and communication overhead. Our performance study in Section 4 shows

that in most cases, no multi-hop proxy is used at all. Therefore, there is no need for a node to proactively maintain up-to-date multi-hop connectivity information. In the rare situation when a node does need to use a multi-hop proxy to deliver a key to a neighbor, it performs an operation similar to expanding ring search to find a proxy.

Authenticated Node Revocation To revoke a node, the key server broadcasts a notification in the network to initiate a group rekeying. The notification must be authenticated so that compromised nodes cannot revoke a legitimate node or spread malicious packets that could lead to inconsistency in our schemes.

We employ TESLA [14] for broadcast authentication in our protocol due to its efficiency and tolerance to packet loss. Let u be the node being revoked, M be the id of the non-compromised key that is possessed by the maximum number of the remaining nodes, and k_i^T be the to-be-disclosed TESLA key. To revoke node u from the network, the key server broadcasts

$$KeyServer \rightarrow * : u, M, f_{k'_g}(0), MAC(k_i^T, u|M|f_{k'_g}(0)). \quad (1)$$

We refer to $f_{k'_g}(0)$ as the *verification* key because it enables a node to verify the authenticity of the group key k'_g that it will receive later. The key server distributes the MAC key k_i^T after one TESLA interval[1]. A node receiving the above node revocation message and the MAC key K_i^T verifies the message using TESLA. It will store the verification key $f_{k'_g}(0)$ temporarily if the verification is successful.

2.4.3. Discussion We now discuss some additional issues that arise in the deployment of our key management protocol.

Node Additions Additional nodes may be added into the system even after the *key pre-distribution* phase discussed in Section 2.4.1 is complete. For example, the key server may introduce new nodes into the system to compensate for revoked nodes. To add a node u into the system, the key server first determines its key set R_u based on its node id. Then it loads node u with the current version of R_u and the current group key that allows u to communicate with other nodes in the network. Depending on the application under consideration, if a group rekeying is needed to prevent a new node to understand the earlier communication, the key server can simply broadcast a message instructing every node to update the group key k_g to $k'_g = f_{k_g}(0)$.

Key Distribution Since the main function of a network is distributing data, not distributing keys, it is reasonable to assume that an infrastructure, such as a multicast delivery tree [16] or mesh [11], or a broadcast tree [8], exists for data communication. None of these protocols are based on flooding for data distribution. Indeed, these protocols provide loop-free routing paths and ensure that every node receives every data packet only once, although they do involve bandwidth overhead in constructing and maintaining the delivery infrastructure. GKMPAN does not construct its own specific delivery infrastructure, but uses the underlying infrastructure directly. Thus, GKMPAN neither has the overhead of constructing and maintaining its own delivery infrastructure, nor introduces redundant messages in key distribution.

Packet Loss and Network Partitions When a node experiences packet losses due to unreliable transmission links or temporary network partition, it may miss one or more group rekeying events. Thus, when it reconnects to the rest of the network, it will detect that the others are using a newer version of the group key. Our rekeying scheme has the attractive feature of *partial statelessness*, which makes it robust in this situation. It is stateless because a node can obtain the current group key securely from other nodes even if it has missed some group rekeying events, as long as it has one non-compromised KEK that allows it to establish a secure channel with the others. Recall that in step 4 of a group rekeying operation, a non-compromised key is updated by applying a pseudo-random function f to it. Thus a node can update its non-compromised KEKs to their current versions independently once it knows how many group rekeying events it has missed (note that every key has a key version field), and then use them to establish secure channels with others. However, the statelessness in GKMPAN is partial because a node can only miss a *limited* number of group rekeying events. When its key subset is completely covered by that of the coalition of the nodes being revoked (see Section 3 for the covering probability), it will not have any KEKs to establish a secure path to its neighbors to obtain the current group key. In this case, it could request the key server on the fly for necessary information to update its keys, using its pre-loaded individual key to secure their communication. In the extended version of this paper [22], we present several mechanisms, including a local recovery algorithm, to address the reliability issue for key delivery.

Networks with Non-member Nodes In the previous discussion, we assumed that all nodes in the network were members of the group under consideration. We now discuss how GKMPAN can be extended to networks where only a fraction of nodes are part of the multicast group.

In multicast routing protocols for mobile ad hoc networks such as ODMRP [11] and the multicast extension of AODV [16], one or several non-member nodes may be involved in forwarding data packets for the group members that are not directly neighboring. In the case of group key management, the group members and some non-members form a multicast delivery tree [16] or mesh [11] with the key

1 We can deploy the variant of TESLA proposed in [13] instead because it allows a receiver to verify the received messages immediately, thus preventing possible resource consumption attacks.

server of the group as the source. The member nodes can be considered to form an overlay network on top of the underlying multicast delivery tree, which includes both members and non-members.

Similarly, GKMPAN can use an overlay network formed on top of a multicast delivery tree for the purposes of secure key distribution. In GKMPAN, a node will need to know the ids of its logical neighbors in the overlay network in order to establish a secure logical paths with them. Currently in ODMRP and AODV a member node does not necessarily know the ids of its neighbors in the overlay network; however, node id information will be available when piggybacked in a JOIN REQUEST, ROUTE REQUEST, or other membership control messages.

In GKMPAN, once member nodes know the ids of its neighbor nodes in the overlay network, they can communicate securely through their logical paths. Although non-member nodes are involved in forwarding the messages for member nodes, they cannot decrypt the messages. We note, however, that a selfish non-member node (just like a compromised member node) could launch refusal-of-service attacks by dropping the keying packets it is supposed to forward. We will discuss how such attacks can be mitigated in Section 3.

3. Security Analysis

In this section, we first analyze the security of our group rekeying scheme, then discuss several other attacks.

3.1. Security of The Group Rekeying Scheme

The correctness of our group rekeying scheme derives from the fact that (i) none of the revoked nodes can generate the new group key k'_g since they do not know k_M, (ii) none of the revoked nodes can obtain the new group k'_g since k_{im} is transmitted via secure logical paths established with keys not known to any of the revoked members, and (iii) every node can verify the new group key independently using the verification key it receives in the authenticated revocation notice.

Since the status of the system is reinstated to its original setting after every rekeying, we only need to consider the possible security issues that arise during a single rekeying operation. One security issue is resulted from a batched rekeying where multiple nodes are revoked simultaneously (e.g., multiple soldiers are captured simultaneously in a combat). If the number of simultaneously compromised nodes is large enough, their coalition may have keys that completely cover the key set of a legitimate node. This implies that a legitimate node will no longer have any keys that it can use to establish any logical paths to other nodes for obtaining a new group key. This node is therefore excluded from the network *innocently*, even though it is still a legitimate node.

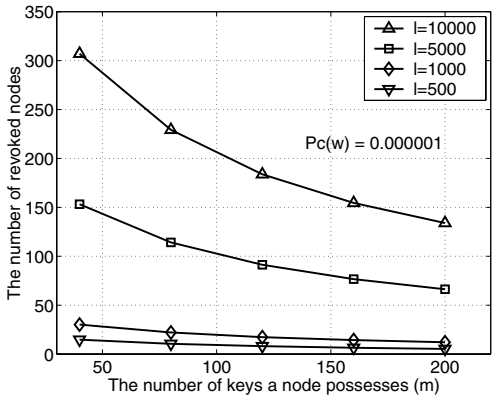

Figure 1. The number of colluding nodes (i.e., w_0) the scheme can tolerate by choosing different m and l pairs, given $p_c(w) = 10^{-6}$

Recall in our key predistribution scheme each key in the key pool has a probability of m/l to be chosen by each node. Consider a key K in the key pool and any coalition of w nodes, the probability p_c that key K is contained in the union of the key sets of the w nodes is $p_c = (1 - (1 - \frac{m}{l})^w)$. Therefore, $p_c(w)$, the probability that all the m keys of a legitimate node are completely covered by that of w colluding revoked nodes, is

$$p_c(w) = (1 - (1 - \frac{m}{l})^{w})^{m}. \quad (2)$$

From this equation, we know that by varying m and l we can obtain a desired security level. Fig. 1 depicts the number of colluding nodes (denoted as w_0) that the scheme can tolerate for a desired $p_c(w) = 10^{-6}$ for different m and l pairs. We observe that w_0 decreases with m but increases with l, and we can choose the appropriate l and m to receive the desired security. For example, in a battlefield, it is usually enough to assume at most 100 soldiers are captured simultaneously. We can choose $m = 100$ and $l = 5000$ according to the figure. That is, for a group size of $1,000,000$, only one innocent node will be excluded from the network when 100 nodes are compromised simultaneously.

In Fig. 2, we further show that the number of innocently excluded nodes grows at a rapidly increasing speed with the number of compromised nodes. Therefore, previously proposed schemes [4, 3, 23] are not scalable with respect to the number of compromised nodes allowed in the system, due to the lack of a key updating mechanism that prevents nodes revoked during different rekeying events from colluding. In our protocol, only the simultaneously revoked nodes can collude. Thus, although our protocol is proposed

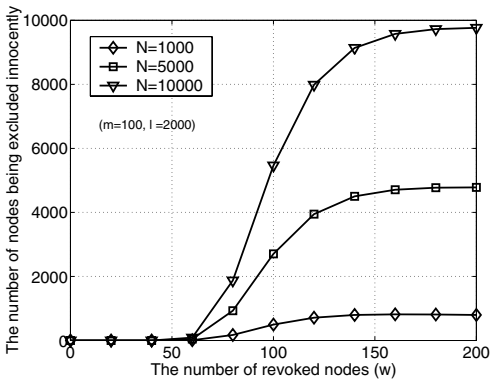

Figure 2. The expected number of nodes being excluded from a network of size N innocently, as a function of the number of revoked nodes w

for group rekeying, it could also be used as an approach for key updating in their schemes.

3.2. Other Security Attacks

We assumed that an attacker may eavesdrop on all traffic, inject packets or replay older packets. Because the key server authenticates all the rekeying messages by TESLA [13], no nodes can inject any fake rekeying messages into the network or modify any rekeying messages they are forwarding while impersonating the key server. That is, malicious nodes cannot cause other nodes to accept wrong group keys. Because time-stamp information is also embedded in every TESLA key, the attacker cannot replay older rekeying packets.

Possessing the keys of revoked nodes normally does not help an attacker launch refusal-of-service attacks, because the revoked nodes are not part of the multicast delivery tree any longer – they do not have any valid keys to establish secure logical paths to other nodes. Moreover, the worst situation caused by refusal-of-service attacks is equivalent to that due to packet losses or network partitions, which our protocol can handle well due to its partial statelessness property. We note that the mobility of nodes helps relieve this attack as well.

4. Performance Analysis

We now study the performance of our rekeying protocol GKMPAN. We analyze the communication cost of GKMPAN and discuss the tradeoff between performance, security and storage cost. We do not consider the computational cost because GKMPAN only involves a few inexpensive symmetric key operations and/or pseudo random function evaluations during every group rekeying.

4.1. Communication Cost

The communication cost of a group rekeying operation is composed of the cost for broadcasting a node revocation notice and the cost for secure group key distribution. The revocation notice includes the id(s) of the node(s) being revoked, a verification key and a MAC of the notice, and the message is usually very small unless the number of nodes being revoked is very large. Note that a revocation notice is necessary no matter what rekeying scheme is being used.

The communication cost for secure key distribution is determined by the numbers and types of logical paths that exist between two nodes in the network. The number of logical paths between two nodes mainly depends on (i) the parameters l and m (ii) the number of nodes being revoked at a batch.

Impact of Probabilistic Key Sharing Parameters For a given l and m, p_s, the probability of directly sharing at least one key between any two nodes initially (i.e.,prior to any node revocations) is

$$p_s = 1 - \frac{\binom{l}{m} \cdot \binom{l-m}{m}}{\binom{l}{m}^2} = 1 - \frac{\binom{l-m}{m}}{\binom{l}{m}}. \qquad (3)$$

Because the probability that a key is picked by both the nodes is $(\frac{m}{l})^2$, z_s, the expected number of keys shared between any two nodes initially (i.e., the number of *direct* paths between them) is $z_s = l \cdot (\frac{m}{l})^2 = \frac{m^2}{l}$.

From the above analysis, we see that for a given l, both p_s and z_s increase with m, while both p_s and z_s decrease with l for a given m. Also, we can see that m has a larger effect on z_s than l has. Thus, we can use the equations above to select l and m so that the probability of existing *direct* paths between two nodes is large enough. For example, to obtain $p_s = 99.5\%$, we can choose $l = 2000$ and $m = 100$ ($z_s \approx 5$). In this case, only 0.5% nodes have to receive the group key through an indirect path. Thus, the average transmission cost per node is close to one key.

Impact of Number of Nodes Being Revoked Our analysis above shows how the parameters m and l affect the *initial* number of logical paths between any two nodes. However, the keys known to the nodes being revoked as a batch cannot be used during the current rekeying event. As a result, both p_s and z_s decrease with the number of revoked nodes.

Let $p_s(w)$ be the probability that two nodes share at least one key when w nodes are revoked simultaneously. We use the following analysis to compute $p_s(w)$. Consider a key k. The probability p_1 that k is chosen by both nodes is $(m/l)^2$, and the probability p_2 that k is not chosen by any of the w

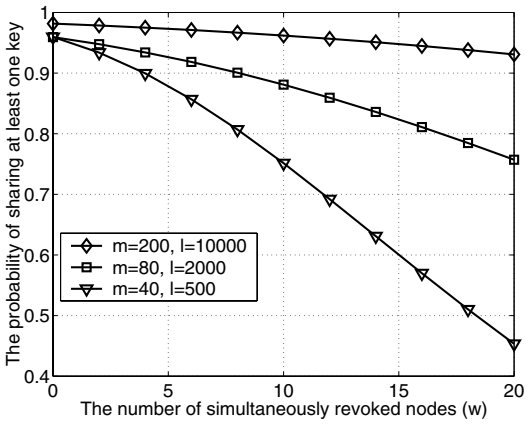

Figure 3. The probability of sharing at least one key between two nodes when w nodes are revoked simultaneously

compromised nodes is $(1 - m/l)^w$. Therefore, the probability p_0 that k is "safe" is $p_0 = p_1 p_2$. The probability $p_s(w)$ is the same as the probability that at least one of the l keys in the key pool is "safe" to both nodes when w nodes are revoked. Therefore, we have

$$p_s(w) = 1 - (1 - (m/l)^2(1 - m/l)^w)^l. \quad (4)$$

Similarly, $z_s(w)$, the expected number of shared keys between two nodes when w nodes are revoked is

$$z_s(w) = l(m/l)^2(1 - m/l)^w. \quad (5)$$

In Fig. 3 we plot $p_s(w)$ as a function of w. The figure clearly indicates that $p_s(w)$ decreases with w. Hence, in order to deliver the group key to a downstream node, a node might have to resort to using a one-hop proxy or even a multi-hop proxy. As a result, the cost of group rekeying increases with w. The figures also shows that it is possible to achieve a desirable $p_s(w)$ by choosing appropriate m and l. For example, when $m = 40, l = 500$, $p_s(10) = 75\%$; whereas $p_s(10) = 96\%$ when $m = 200$ and $l = 10000$. Thus, in the latter case the communication cost of a small batch rekeying is almost as low as an individual rekeying.

A Rough Comparison with LKH We now compare the communication cost of GKMPAN with that of LKH [19]. The reason for comparing our protocol with LKH is that it illustrates the differences in communication cost between a protocol that was designed for a wired environment as opposed to a protocol that is geared towards wireless ad hoc networks. The group rekeying scheme for ad hoc networks in [10] shows that it is possible to reduce the cost of the original LKH scheme by $15\% \sim 37\%$ (although it may incur a larger overhead in some scenarios) by mapping the physical locations of the members to the logical key tree in LKH for a static network. Because the performance overhead in [10] is of the same order as that of the original LKH scheme, we can also see the comparative performance of GKMPAN with respect to the protocol described in [10]. Note that we do not consider reliable key delivery in the comparison, which actually biases the comparison in favor of LKH since it is a stateful protocol.

As discussed in Section 2.4.3, GKMPAN uses the underlying delivery infrastructure for key distribution. In the simulation study, we use an underlying spanning tree for delivering keys in both the protocols. Both the key server and the nodes are randomly distributed in a fixed space of 1000×1000 square units and the transmission range of a node is 75 units. We consider a static network, which corresponds to a snapshot of the ad hoc network at the time of a rekeying event, instead of a mobile network. This is because group rekeying usually takes a very short time relative to mobile node velocities.

The metrics of interest are the average numbers of keys a node transmits and receives respectively in every rekeying. We use the method of independent replications for our simulation. All the results have 95% confidence intervals that are within 5% of the reported values. In Fig. 4 we compare the communication cost of GKMPAN and LKH by varying w, the number of nodes revoked as a batch. The communication cost does not include the cost of the node revocation notice because both LKH and GKMPAN incur this cost. The group size is $N = 1024$. In GKMPAN, $m = 40$ and $l = 500$. We can observe that the communication cost for LKH is much higher than that of GKMPAN. In GKMPAN the average communication cost per node is approximately one key and is almost independent of L. In LKH, both the number of keys a node transmits (T_n) and the number of keys a node receives (R_n) increase with L. Here $R_n > T_n$ because all the nodes receive the updated keys whereas only non-leaf nodes in the broadcast tree rebroadcast the keys. Therefore, GKMPAN greatly outperforms LKH for individual or small batch rekeying in ad hoc networks. In Fig. 5, we observe a similar result when varying the group size, N.

4.2. Security, Performance and Storage Tradeoffs

As we observed above, to increase the number of *direct* logical paths between two nodes, it is necessary to increase m or decrease l. Further, increasing m has a larger impact on z_s. However, from the viewpoint of storage, a smaller m is more desirable. Moreover, as we showed in Fig. 1, a smaller m and a larger l could increase the security of our schemes. Due to these conflicting requirements, the parameters m and l should be selected based on the application under consideration.

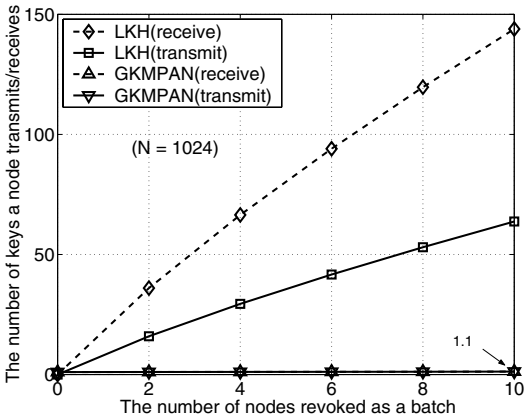

Figure 4. The impact of the number of nodes being revoked as a batch on the communication cost of LKH and GKMPAN respectively.

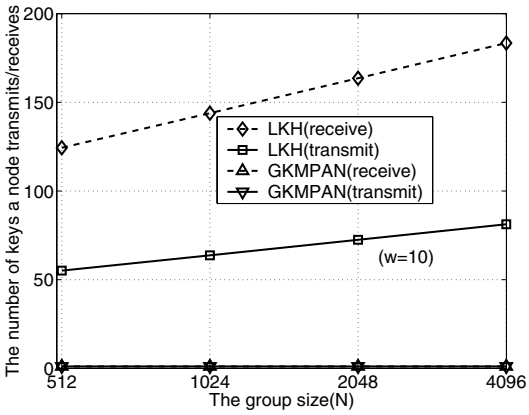

Figure 5. The impact of the group size on on the communication cost of LKH and GKMPAN respectively

A Comprehensive Example Suppose that a node has space for 200 keys, i.e., $m = 200$. The key server chooses $l = 10000$. According to eqn. 3, we have $p_s = 98.3\%$, i.e., the probability that two nodes share *direct* paths is 98.3%. Let the desired security level be $p_c(w) = 10^{-6}$. According to eqn. 2, we have $w = 134$, i.e., the probability that the coalition of 134 nodes have keys to cover a legitimate node is about 10^{-6}. When 50 nodes are simultaneously revoked, the fraction of direct, indirect paths involving a one-hop proxy, and indirect paths involving a two-hop proxy are 76.7%, 22.7% and 0.6% respectively. Our simulation shows that on average a node needs to receive and transmit 1.2 keys.

5. Related Work

Group Rekeying Schemes Group rekeying has been extensively studied in the context of secure multicast in wired networks. The rekeying schemes can be categorized into stateful and stateless protocols. The stateful class of protocols includes several protocols based upon the use of logical key trees, e.g., LKH [19], OFT [2], ELK [15]. In these protocols, the key server uses key encryption keys that were transmitted to members during previous rekeying operations to encrypt the keys that are transmitted in the current rekeying operation. Thus, a member must have received all the key encryption keys of interest in all the previous rekey operations; otherwise, it will not be able to decode the new (group) key. Adding redundancy in key distribution [18] will not solve the problem completely due to high packet loss rates or network partitions. *Stateless* group rekeying protocols [9, 12, 17] form the second class of rekey protocols. In these protocols, a legitimate user only needs to receive the keys of interest in the current rekey operation to decode the current group key. The stateless feature makes these protocol very attractive for ad hoc networks. However, these protocols have much higher communication overhead than the stateful protocols. GKMPAN differs with the above schemes mainly in two respects. First, it provides *partial* statelessness, that is, a node can miss a certain number of group rekeyings with the need of asking the key server for retransmission. Second, GKMPAN has much smaller per node transmission cost than the other schemes.

Other Key Management Schemes Basagni et al [1] discuss a rekeying scheme for periodically updating the group-wide data encryption key in a stationary sensor network. However, they assume that the sensor nodes are tamper-proof, and do not address the issue of rekeying on node compromise. Zhu et al [21] propose a group key management scheme for stationary sensor networks. Kaya et al [7] present a group key management scheme for ad hoc networks based on public-key techniques. GKMPAN, using symmetric-key techniques only, is designed for mobile ad hoc networks and does not require tamper-resistant nodes.

Eschenauer and Gligor [4] present an efficient key management scheme for sensor networks based on probabilistic key predeployment. Chan et al [3] and Zhu et al [23] extend this scheme and present new mechanisms for pairwise key establishment. GKMPAN also uses the probabilistic key predeployment technique as the underlying means to establish secure channels between nodes. However, in these schemes, the predeployed keys are used for encrypting all communications between nodes; there is no group key. In contrast, we propose to use the predeployed keys only as KEKs for securely distributing a group key to the nodes in the network while using the group key for securing group data communications. Thus, GKMPAN incurs

much smaller communication and computational overhead in group communication. Last but not least, those schemes do not update any compromised keys, while GKMPAN includes an efficient mechanism to update the keys known to revoked nodes.

6. Conclusions

In this paper, we have presented GKMPAN, a scalable and efficient group key management protocol for ad hoc networks. The main component of GKMPAN is a novel group rekeying protocol that has the following properties:

- It is efficient – it relies only on symmetric key cryptography and the computational and communication cost of group rekeying are distributed among the nodes in the network.
- It is scalable – the storage requirements per node are independent of the size of the network.
- It is partially stateless – a node can decode the current group key even if it has missed a limited number of previous group rekeying operations.

Our protocol is based on a probabilistic key sharing scheme that can be parameterized to meet the appropriate levels of security and performance for the application under consideration.

Finally, we refer the interested reader to an extended version of this paper that includes rigorous security proofs for our rekeying scheme, a description of an extended version of our rekeying scheme that relaxes some of the assumptions made in this paper such as immediate compromise detection, and a discussion of the implementation of GKMPAN on a sensor network testbed [22].

Acknowledgments

We thank the anonymous reviewers for their valuable comments.

References

[1] S. Basagni, K. Herrin, E. Rosti, D. Bruschi. Secure Pebblenets. In Proc. of MobiHoc 2001.

[2] D. Balenson, D. McGrew, and A. Sherman. Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization. IETF Internet draft (work in progress), August 2000.

[3] H. Chan, A. Perrig, D. Song. Random Key Predistribution Schemes for Sensor Networks. In Proc. of the IEEE Security and Privacy Symposim 2003, May 2003.

[4] L. Eschenauer and V. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In Proc. of ACM CCS 2002.

[5] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. Journal of the ACM, Vol. 33, No. 4, 1986, pp 210-217.

[6] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11, 1997

[7] T. Kaya, G. Lin, G. Noubir, A. Yilmaz. Secure Multicast Groups on Ad Hoc Networks. In Proc. of ACM Workshop SASN'03.

[8] H. Lim and C. Kim, Multicast tree construction and flooding in wireless ad hoc networks, in Proc. of ACM MSWIM 2000.

[9] D. Liu, P. Ning, and K. Sun. Efficient Self-Healing Group Key Distribution with Revocation Capability. In Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03), October, 2003.

[10] L. Lazos and R. Poovendran. Energy-Aware Secure Multicast Communication in Ad-hoc Networks Using Geographic Location Information. In Proc. of IEEE ICASSP'03, Hong Kong, China, April, 2003.

[11] S. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks. ACM/Kluwer Mobile Networks and Applications, 2000.

[12] D. Naor, M. Naor, and J.Lotspiech. Revocation and tracing schemes for stateless receivers. In Advances in cryptology - CRYPTO 2001, Santa Barbara, CA, Aug. 2001.

[13] A. Perrig, R. Canetti, D. Song, and J. Tygar. Efficient and secuire source authentication for multicast. In Proc. of NDSS'01, Feb. 2001.

[14] A. Perrig, R. Canetti, J. Tygar, D. Song. Efficient authentication and signing of multicast streams over lossy channels. In Proc. of IEEE Security and Privacy. May 2000.

[15] A. Perrig, D. Song, D. Tygar. ELK, a new protocol for efficient large-group key distribution. In Proc. of the IEEE Security and Privacy Symposim 2001, May 2001.

[16] E. Royer and C. Perkins. Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol. In Proc. of MobiCom '99, Seattle, WA, Aug. 1999.

[17] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin and D. Dean. Self-Healing Key Distribution with Revocation. In Proc. of the IEEE Security and Privacy, 2002.

[18] S. Setia, S. Zhu and S. Jajodia. A Comparative Performance Analysis of Reliable Group Rekey Transport Protocols for Secure Multicast. In Performance Evaluation, special issue on Proceedings of Performance 2002, Rome, Italy, Sept 2002.

[19] C. Wong, M. Gouda,S. Lam. Secure Group Communication Using Key Graphs. In Proc. Of SIGCOMM'98, 1998.

[20] URL: http://www.xbow.com

[21] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks In Proc. of ACM CCS '03, October, 2003.

[22] S. Zhu, S. Setia, S. Xu, and S. Jajodia. GKMPAN: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks Technical report ISE-TR-02-01, Feb. 2004. Http://mason.gmu.edu/∼szhu1/gkmpan-techreport.pdf.

[23] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing Pairwise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach. In Proc. of ICNP'03.