The Effects of Probabilistic Key Management on Secure Routing in Sensor Networks

Patrick Traynor, Guohong Cao and Tom La Porta
Penn State University

Abstract— Secure data dissemination is a necessary and an extremely important component of ad hoc sensor networks and has been the topic of a large body of literature over the past few years. A variety of schemes have been proposed in order to ensure that data is delivered through these networks while maintaining message authenticity, integrity and, if so desired, privacy. The majority approaches applied to ad hoc networks assume the presence of either public or pre-established symmetric keys. This assumption, however, is not realistic for sensor networks. In this paper, we discuss the use of probabilistic symmetric-key management schemes and the ways in which their deployment specifically affects the ability of sensor nodes to optimally route packets in a secure setting. While numerous papers have advocated such an approach, none have investigated the details of such an implementation. Specifically, we contrast pre-establishing symmetric keys with neighboring nodes to a completely reactive approach of instituting secure relationships. Through simulation, we quantify the consequences of the application of these two methods on a number of scenarios requiring secure hop-by-hop routing in sensor networks.

Index Terms— Secure Routing, Probabilistic Key Distribution, Key Management, Sensor Networks.

I. INTRODUCTION

Providing secure operation of ad hoc wireless sensor networks is a much more challenging task than with their traditional wired counterparts. Nodes in wired networks are able to leverage considerable resources in terms of power, processing ability and positive human/administrator interaction along with centralized, trusted servers. Nodes of ad hoc sensor networks typically lack the aforementioned means. To make matters worse, the placement of these systems is often in environments which are inherently risky, such as a theatre of war. Therefore, attempts to implement security cannot assume that nodes possess physical safety. Accordingly, security solutions for sensor network applications must typically be distributed, robust and as efficient as possible.

Many similar problems have been addressed when designing security solutions for mobile ad hoc networks (MANETs). In contrast to MANETs, sensor networks typically consist of a large number (hundreds to a few thousand [1]) of very simple nodes. Commercially available sensors, such as the Berkeley MICA2 mote, are characterized by their limited processing capability (8-bit, 4MHz processor), tiny memory (128 KB program memory) and small size [6]. Additionally, sensor nodes are expected to operate without direct human intervention from the moment they are deployed. It is because of these extremely minimal characteristics that sensor networks are an even more difficult subset of wireless systems than MANETs and therefore require special consideration when attempting to apply security solutions.

One of the most important challenges facing ad hoc sensor networks is that of secure data dissemination. The focus of a large pool of literature in recent years, secure data dissemination is absolutely essential to network fidelity as nodes are completely reliant upon their neighbors to forward packets to their intended destinations. In a setting where individuals must be even more skeptical of the activities of their neighbors for all of the abovementioned reasons, simply trusting that an adjacent node is performing the expected ameliorative operations on our data would be naïve. While a number of schemes have been suggested to make such neighbor interaction more secure in a MANET, the majority of works have failed to truly translate to a sensor network environment as the key management strategies implemented by those methods are either vague or infeasible.

A well received solution to the issue of key management in sensor networks that has been extended by a number of researchers [5][7][8][9][14][20][25] is to distribute a certain number of randomly selected keys in each of the nodes throughout the network. Using this approach, one can achieve a known probability of connectivity within a network while minimizing the resources necessary to implement the security system. The specific consequences of this approach when applied to routing and data dissemination, however, have not yet been explored. It is for these reasons that we investigate this method of key management with secure routing schemes from MANETs in order to provide secure data dissemination for sensor networks.

The remainder of the paper is organized as follows: Section II discusses specific related works in the fields of both secure routing and probabilistic key management. Section III presents the protocol used to establish pair-wise secure relationships. Section IV introduces and discusses a number of network scenarios and the inherent costs associated with each. Section V presents and analyzes performance data from simulations of the scenarios in Section IV. Finally, concluding remarks are offered in Section VI.

II. RELATED WORK

Before we discuss the application of a probabilistic keying scheme to data dissemination and secure routing mechanisms, it is necessary to understand the previous work done in each of these areas.

Specifically designed for sensor networks, data dissemination protocols, such as Directed Diffusion [13], rely upon multi-hop forwarding to deliver data. Such protocols operate in a data centric manner, whereby nodes in the network are addressed by the content or data they provide and not by an IP address as is done in traditional networks. In order to retrieve information, a data sink injects information requests in the form of “interests” and waits for responses to be forwarded by intermediate nodes toward him/her. In order to secure such a protocol, we look to the work already conducted in MANETs for inspiration.

Some approaches to secure routing in MANETs have suggested that a cryptographic interaction between the endpoints of communication is necessary to implement secure data exchange. For example, in [15] the source and destination nodes share each others' public keys. When the DSR route request (RREQ) launched by the source arrives at the destination, the target node signs the list of nodes used to reach it with its private key. Upon receiving the response, the sender can be sure that the RREQ indeed did reach the desired node and secure interaction can begin.

A number of other papers in this domain have proposed the use of neighbor-based authentication. The authors of [19] advocate the ARAN protocol, which can be run in conjunction with either AODV and DSR. When an intermediary node in the route receives an RREQ, it examines the certificate and contents encrypted by the private key of the previous hop. If the decrypted value correctly corresponds to the expected value of the packet, the current node then removes the previous certificate and adds its own before encrypting and forwarding the packet on to its next hop.

Ariadne, as presented in [10][12], combines the previous two approaches by having both the target and intermediary participants take part in the authentication of routing data. Before sending the RREQ, the source computes an HMAC using a key k_{SD} shared between itself and destination. During route discovery, each node along the source-destination path authenticates routing information with its Tesla [18] key. The destination then buffers the packet until the requisite amount of time passes for each intermediary to release its Tesla key before processing the packet. Zapata and Asokan in [24] take a similar methodology by combining the hop-by-hop and end point approaches, but rely upon public keys in order to perform hashing and signatures for routing packets citing the difficulties with time synchronization inherent to the release of Tesla keys.

Capkun and Hubaux are some of the first to discuss the possibility of secure routing in a wireless setting without having a secure relationship between all nodes in [4]. Here, the authors attempt to reduce the amount of information that must be stored in each node by recognizing that each node is likely to only communicate end-to-end with a small subset of the total nodes in the network. Through key establishment methods discussed in [10], they argue that Ariadne can operate with an initially incomplete set of security associations.

The above solutions, and a number of additional approaches including [2] and [21], make simplifying assumptions when the issue of key management is raised. The majority of these schemes default to the presence and capability of nodes to use public-key cryptography to accomplish their goals; however, such approaches are not possible in sensor networks due to the previously discussed limitations of node capabilities. Those papers not specifically requiring the non-repudiation characteristics inherent to public-key methods instead typically state that symmetric keys already exist between nodes but do not discuss the method in which those keys have been established.

The two major exceptions are those schemes suggesting the possibility of using Tesla keys [18] and a brief discussion of the potential for the use of probabilistic keying in [4]. The Tesla key-based approach has been discussed in conjunction with sensor networks in [17]; however, its reliance upon a centralized base station for the calculation and distribution of keys effectively limits the size of those networks. Indeed, any solution applied to this problem must be as robust and scalable as possible.

The most ideal key management approach would be for each of the n nodes in the network to store $n-1$ unique keys, one for each peer in the network. While providing perfect resilience to node compromise and ensuring that all nodes can communicate directly, this scheme incurs a growth rate of $O(n^2)$ and therefore does not scale to large networks. Other methods that attempt to assign keys specific to an a priori position, fail to take into account the potential for mobility or insertion of new nodes, and are not robust against potential node misplacement.

The probabilistic distribution of keys, which has been proposed and extended by several researchers [9][5][7][8][14][20][25], distributes a certain number of randomly selected keys in each of the nodes *a priori* to network deployment. Using this scheme, it becomes possible achieve a known probability of connectivity within a network.

The seminal work in this field was presented by Eschenauer and Gligor in [9]. In this scheme, a large pool of P keys is generated, from which k are randomly selected, without replacement, for each sensor node. Two nodes may communicate to directly establish a session key if they have a key match. The probability that two nodes with the same number of random keys, k, share at least one key is:

$$P(Conn) = 1 - \frac{((P-k)!)^2}{P!(P-2k)!} \qquad (1) \quad [9]$$

When attempting to determine whether or not a key is shared between a pair of neighbors, nodes broadcast a plaintext listing of their key identifiers. These identifiers are randomly assigned to each of the keys before nodes are placed in the field and therefore do not give attackers any additional information about the key values themselves. If a neighbor has a key corresponding to one of the broadcasted identifiers, it answers the source node with a challenge-response message. Nodes not directly sharing keys can establish session keys via indirection through commonly trusted neighbors.

Extensions to this work include the requirement that nodes share more than one key in order to communicate securely in [5] and placing different number of keys in nodes depending on their capabilities and missions in [20].

The advantages of such probabilistic approaches are numerous. Most importantly, they allow for secure communications to be possible in resource constrained platforms while using very little memory for key storage. In a setting with exceedingly limited resources, where hop-by-hop authentication for secure routing is a desirable goal, such a keying scheme is particularly attractive.

We therefore investigate how a hop-by-hop secure routing scheme performs in sensor networks in the presence of probabilistic keying. We examine two approaches for establishing session keys. In the first, nodes proactively establish keys with their direct neighbors; we call this the proactive neighbor approach (PNA). In the second, nodes only establish keys when they are required, for example to authenticate a message; we call this reactive keying (RK). We examine both cases intuitively and through simulation. Before discussing these issues, however, we first define in detail the protocol used for establishing pair-wise keys between neighbors.

III. SHARED-KEY DISCOVERY PROTOCOL

While a number of the previously mentioned probabilistic keying schemes [9][5][20] make use of the broadcasting of key identifiers in order to establish pair-wise keys, none of them explicitly define the protocol. In this section we describe the protocol used in our system for establishing session keys.

The first step of the key discovery protocol is the *direct key match* phase. In this phase, nodes desiring to establish a secure relationship with their neighbors broadcast a set of identifiers corresponding to their pre-deployed keys. We assume that nodes know their direct neighbors by exchanging periodic '*hello'* messages. The source node then waits for challenge-responses from its neighbors with which keys are shared. During the challenge-response, session keys are established between nodes with a key match.

Once the direct key match phase concludes, nodes will have session keys with neighbors for which an a priori distributed key is shared, but will still be without session keys for some portion of their neighbors. To establish keys with these nodes the *indirect key match phase* commences. Determining when to begin the indirect key match phase is difficult. Methods for determining this moment include the use of key identifier broadcast NACKs, multiple retransmissions of the initial key identifiers in case nodes were unable to hear previous attempts (given difficulties including general collisions and the hidden terminal problem) and the use of a help request after a timer has expired.

Requiring sensor nodes to respond to every request for keys (even when they do not have a shared key with a source) forces nodes to needlessly spend their most precious commodity - energy in the form of transmissions. Additionally, if a node did not hear the initial transmission of key identifiers, it will never send a NACK. Attempting to retransmit the key identifiers multiple times in anticipation of corrupted broadcasts is also wasteful of resources. It is quite possible that every node heard the initial broadcast and legitimately has not responded because no shared key exists between the two parties.

We therefore default to the use of timers for the release of an *Indirection Key Message* (IKM). The primary advantage to this approach is that it eliminates the need for both source and neighbor nodes to send potentially wasteful messages. Additionally, in a setting where most neighbors have a reduced number of keys when compared to a few more powerful nodes [20], the reliance upon indirection is therefore increased and the use of a timer allows for the quickest request for keying assistance.

After the expiration of the timer, a node desiring a secure relationship with a node with which it was unable to establish a key in the direct key match phase launches an IKM including the node IDs with which it desires to establish keys. Nodes that already have a secure relationship with the source can use their secure relationships with other nodes on the transmitted list to help establish a session key.

The complete protocol, including both the direct and indirect key match phases, is defined below:

Notations

- *s,u,v* are the source, indirection assisting neighbor and the desired target, respectively.
- ID_x is the key identifier associated with key x where $x \in \{0,1,\ldots,P-1\}$.
- N_m is a node with where $0 \le m \le \#\ nodes-1$
- n is a nonce.
- $k_{x,y}$ is a session key for nodes N_x and N_y.

Protocol Definition

1. s broadcasts a message containing its key identifiers:
$$s \rightarrow [s, ID_0, \ldots, ID_{k-1}]$$
2. Neighbors with a match respond to the broadcast by s:
$$u \rightarrow [s, u, ID_x, k_x(s, u, ID_x, n)]$$
3. s, after the expiration of a timer, broadcasts an IKM message including the identifiers of the nodes with which it could not establish a key in the first phase:
$$s \rightarrow [s, N_1, N_2, \ldots, N_m]$$
4. u, receives the broadcast from step three. Having already established a key with s and v, u transmits the following message with a session key for use between the two:
$$u \rightarrow [s, v, u, k_{u,v}(k_{s,v}, n), k_{s,u}(k_{s,v}, n),$$
$$MAC\ (k_{s,v}, s \mid u \mid v \mid n)]$$
5. v sends a confirmation message to s:
$$v \rightarrow [s, v, k_{s,v}(s, v, n) MAC(k_{s,v}, s \mid v \mid n)]$$

After the fifth step, two nodes previously unable to set up a session key are able to do so. Lastly, if a node attempting to establish keys with all of its neighbors should still lack a key with any of its neighbors, it could simply launch the indirect key match phase using the newly keyed nodes as new hops for indirection. A second approach is to allow neighboring nodes to continue to propagate the help requests. As the original research demonstrates [9] a high probability of establishing keys with all neighbors within one indirection given a large enough number of keys, we will rely upon the first approach should this situation arise in simulations.

The protocol described above is implemented on CrossBow MICA2 Motes using TinyOS. The sensor nodes maintain two data structures. The first is used for nodes to quickly determine if they have matching keys in response to the direct key match broadcast. We assume the pre-deployed keys are sorted in ascending order and maintained in a link list. Our search time to determine if we have a key is then O(log(*k*)). An alternative approach would have been to implement a hash table in the sensors, but this was considered to incur high processing overhead.

The second structure is used to determine which keys match with which node, and is populated during the direct or indirect key match phases. This structure is used to find a particular key for a node for the indirect key matching procedure, or for establishing session keys with nodes that have expired keys. For this we implemented a sorted link list of node identifiers. Each item in the list points to a list of keys that the nodes have in common. We maintain multiple heads on the link list to speed the search. The average search time for these keys is O(log(*h*)) + O(log(*h/n*)), where *h* is the number of heads and *n* is the number of neighbors in the linked list.

The average time required to determine the existence of a key match using the above data structure, given the system characteristics stated in [2], would be approximately 8.6ms. Accordingly, we expect to see that the large majority of time required to execute this probabilistic key management scheme has been relegated to the realm of medium access.

IV. NETWORK SCENARIOS

In terms of key establishment strategies based on a probabilistic distribution, we evaluate both the proactive neighbor approach (PNA) and reactive keying (RK) with direct neighbors. With PNA, upon detection of a previously unseen node, a node proactively establishes a key so that this node is a viable first hop over which a message can be sent. In effect, nodes always maintain keys with their neighbors. PNA has the benefit of faster route establishments and faster local route repairs at the expense of possible wasteful key matching exchanges.

Under RK, nodes only establish secure relationships when there exists a need to do so. This will lead to higher latencies in establishing routes, but will limit unnecessary overhead.

First, we evaluate the performance in an environment in which nodes are not mobile. This provides insight into the overhead of the basic key matching protocol. We then evaluate the performance impact of PNA and RK on a secure version of AODV using a protocol based on ARAN [19] for two types of mobility models – traditional and group mobility.

In the first mobility model nodes move individually. We consider cases in which sessions keys have an infinite lifetime and in which keys expire after a period of time. With PNA, when neighbors have a key that is about to expire, they perform a simple point-to-point handshake to renew the session key. Using RK, nodes that are actively communicating (i.e., have exchanged data within 100 seconds) also renew expiring keys with a simple handshake. Keys established with nodes that are not in active communication expire and are re-established only if required using the key matching protocol described in above.

In the second mobility model we consider a network in which sensor nodes tend to move as a group, as discussed in [16].

V. SIMULATION AND DISCUSSION

We evaluated the effects of probabilistic keying on secure routing using the Network Simulator (ns-2) version 2.26. Each simulation was run over two different field configurations: 200 and 500 nodes in a 500m x 500m test-bed. Nodes were given a maximum transmission range of 75m and used an 802.11 MAC layer. The average number of neighbors for the 200 and 500 node cases was seven and 15, respectively.

Routing was accomplished via the built-in implementation of the AODV protocol. We assert that the hop-by-hop behavior of a protocol such as AODV can be used to give an intuition about the functionality of other protocols including those specific to data dissemination and is therefore an acceptable tool to use for modeling. '*Hello*' messages were launched every two seconds for PNA and once every 100 seconds for RK. When non-keying related data was passed between nodes, 500 byte packets were sent every 0.01 seconds for 0.5 seconds for a total of 50 packets. This transmission was accomplished via CBR traffic over UDP. These data packets were sent once every 100 seconds, after the requisite '*hello*' (for the RK case) and keying messages transpired.

The timer for transmitting IKMs was set to 0.5 seconds. In both the 200 and 500 node density cases, all nodes responded within 0.1 seconds of the initial broadcast of key identifiers. By allowing this extra time, it was our hope that we would allow as many nodes as legitimately possible to establish keys directly.

Initial node placements and their subsequent movements were generated by the included scenario generation utility. Where applicable, mobile nodes move at between zero and five meters per second.

The keys stored in each of the nodes were generated a priori using the library function `random()`, as it is known to create better pseudorandom values than `rand()`. Each key scenario generation was also seeded with the current time. The key pool, *P*, was of size 10,000 and each node received *k*=83 keys as is described for a 50% chance of direct communication in [9].

	Latency (seconds)	
	200 Nodes	500 Nodes
Single Node	1.6119403	4.77569
All Nodes w/Neighbors	99.195217	251.6091
n-1	473.88027	1408.232

Table 1. Average time required to establish keys.

A. Static Network Evaluation

In this scenario we determine the average amount of time required to create secure relationships between nodes in a static network using the key matching protocol described in Section III.

We consider three cases: a single node establishing keys with its neighbors, all nodes establishing keys with all neighbors, and all nodes establishing keys with all other nodes in the network. Although the case in which each node establishes keys with the *n-1* other nodes in the network is not feasible for reasons of scalability, we have included a simulation to demonstrate its further complications and understand its characteristics. The data is shown above in Table 1.

In a network as small as 200 nodes with an average of approximately four and worst case of 11 hops per message, the network requires almost eight minutes to perform complete (*n-1*) pair-wise keying. The more dense case of 500 nodes, with the same average and worst case numbers for hops per message, takes almost 23.5 minutes to complete. It is clear that it is not possible to use complete pair-wise keying if

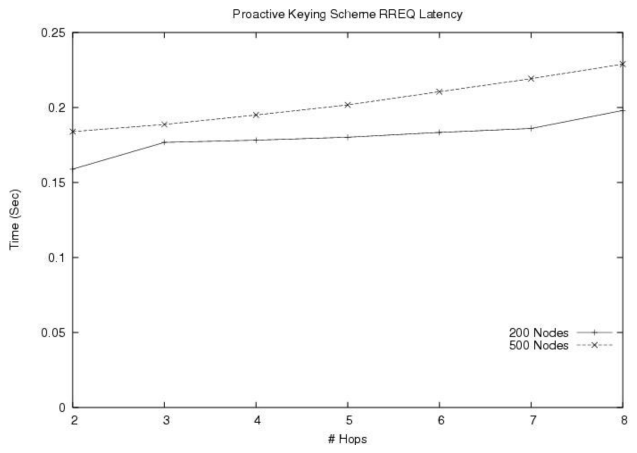

Figure 1: Average delay (sec) for an RREQ to be returned for a varying number of hops with PNA.

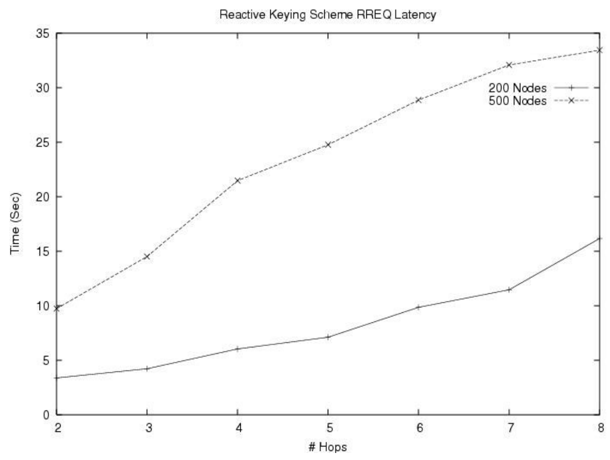

Figure 2: Average delay (sec) for an RREQ to be returned for a varying number of hops with RK.

network initialization time is a concern. Such latencies for key establishment are extremely high and are the result of contention for the air interface. The benefit of this approach should it be implemented on a small scale, however, is that once completed, a wide range of secure data dissemination and routing protocols could be implemented as end-to-end cryptographic operations could be conducted in addition to hop-based measures. Additionally, by placing the expense of keying before the mission-oriented operations of the network take place, the cost of delivering messages end-to-end is equivalent to the examples given in the previously cited literature. The addition of key expirations consequently, would destroy any benefits yielded by such an approach.

The case of all nodes establishing keys with each of their neighbors is the cost of bootstrapping a network with PNA. When bootstrapping the network, each node attempts to begin keying as soon as they become active. In the case where there are 200 nodes this process can take just under two minutes. The scenario containing 500 nodes requires over four minutes. Like the previous *n-1* case, major factor contributing to such long key-establishment times is contention for the air interface which is made more difficult as node density increases.

The case of a single node establishing keys is the cost of adding a new node into a PNA network. It will take 1.6 or 4.8 seconds for such a node to establish keys with its neighbors for the 200 and 500 node cases, respectively. As this new node is the only source requesting new keys, it does not have to deal with sizeable competition for the medium when attempting to establish keys with its neighbors.

The advantage of PNA is that once the bootstrapping phase is complete, hop-by-hop authentication protocols, such as ARAN, will achieve high performance.

The trade-off in terms of routing latency for PNA and RK for a static network is demonstrated in Figures 1 and 2. Figure 1 depicts the routing latency when hop-based authentication is used with AODV after the session keys have been established using PNA.

Figure 2 shows likewise for the initial route requests when using RK. Clearly the latency savings when using PNA are extremely large as this approach places all of its costs up front.

Because this is a static network, once keys are established, they can be easily maintained. With PNA, no new establishments are needed. With RK, new keys will be established until all nodes have keys with all neighbors. Once keys have been established for all nodes, the performance of PNA and RK will converge.

The more interesting cases involving node mobility are described below.

Key Method	Number of Nodes	Infinite Key Lifetime		Finite Key Lifetime	
		Routing+ Key Match	Key Match	Routing + Key Match	Key Match
PNA	**200**	**21.9%**	**19.4**	**40.8**	**39.4**
	500	**53.3**	**52.4**	**63.5**	**62.9**
RK	**200**	**16.6**	**13.8**	**21.9**	**19.4**
	500	**27.5**	**25.4**	**30.6**	**28.6**

Table 2: Recorded values for keying and total overhead for nodes moving at between zero and five m/s.

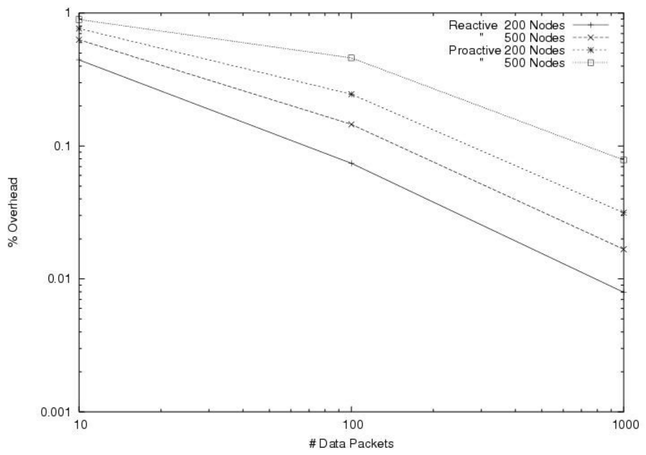

Figure 3: A comparison of traffic intensity versus overhead.

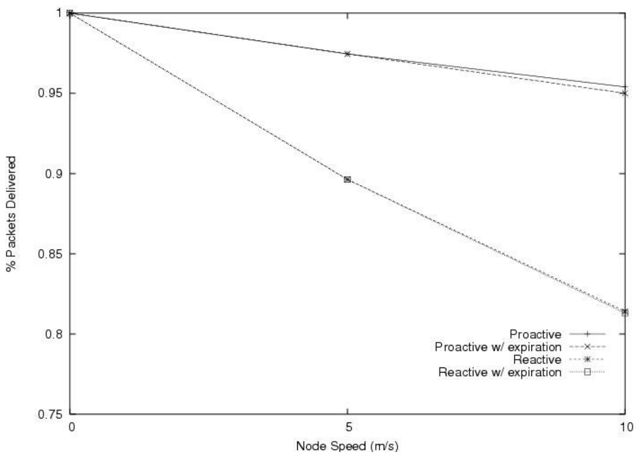

Figure 4: Packet loss ratio for PNA and RK schemes for individual mobility with 200 nodes present in the scenario.

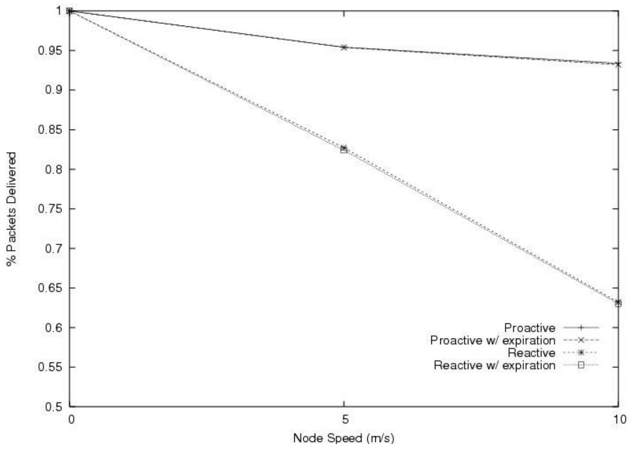

Figure 5: Packet loss ratio for PNA and RK schemes for individual mobility with 500 nodes present in the scenario.

B. Individual Node Mobility

We now examine the performance of PNA and RK in a network in which nodes are individually mobile. While we recognize that this model may not be completely realistic [23], especially in the realm of sensor as compared to ad hoc networks, it is important to conduct these experiments so that a comparable set of results to the aforementioned secure routing schemes can be examined.

We determine routing latency, packet loss ratio, and overhead for the keying strategies. As the curves depicting the latency for establishing routes are very similar to and only slightly higher than the static cases, we will omit them for the sake of space. Unlike the static case where the initial latencies shown in Figures 1 and 2 decrease quickly, route latencies remain high for both the PNA and RK approaches as intermediary nodes are required to establish keys with new neighbors for almost all data transmissions until after the network has reached a steady-state. It is therefore crucial to understand the keying behaviors of nodes at a given time so the consequences of these actions can be used to give deeper knowledge into the abovementioned characteristics.

Table 2 and Figures 3 and 4 show these remaining two of the aforementioned results, respectively, for the steady state operation of PNA and RK for both cases of infinite and finite key lifetimes.

Table 2 shows the overhead incurred with PNA and RK with nodes moving at between . The first column for each protocol shows the overhead including route request exchanges and key establishment messages. The second column under each protocol isolates the overhead of just the key matching.

One interesting occurrence observed in both the PNA and RK schemes is that as nodes move, they establish keys with more nodes. PNA is more aggressive because it establishes keys with every new neighbor it discovers. RK only establishes keys with nodes with which it is exchanging data or route requests. Therefore, in both cases, as the network lifetime increases, the number of new keys being established decreases. This translates into better performance as time progresses until steady-state is reached. Two factors impact the steady-state performance: the mobility pattern and the requirements for key renewal. If nodes are highly mobile, they will tend to have more new neighbors and hence reach steady-state quicker. If keys have an infinite lifetime, once nodes have established secure relationships, they will never need to do so again. In this case performance of both schemes will eventually be the same. However, the frequent establishment of new keys with neighbors, which occurs when key

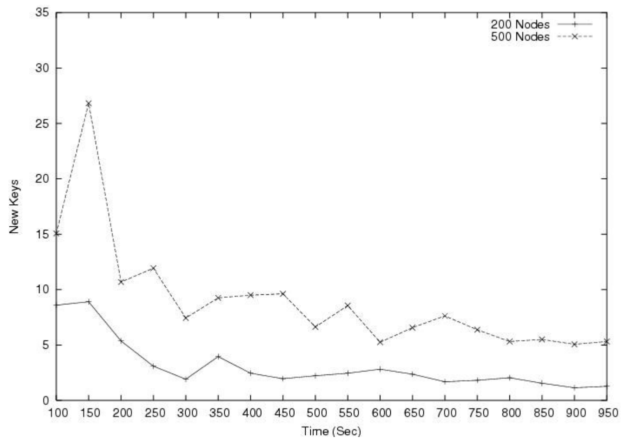

Figure 6: The average number of new keys established per node per 50 seconds with PNA key management.

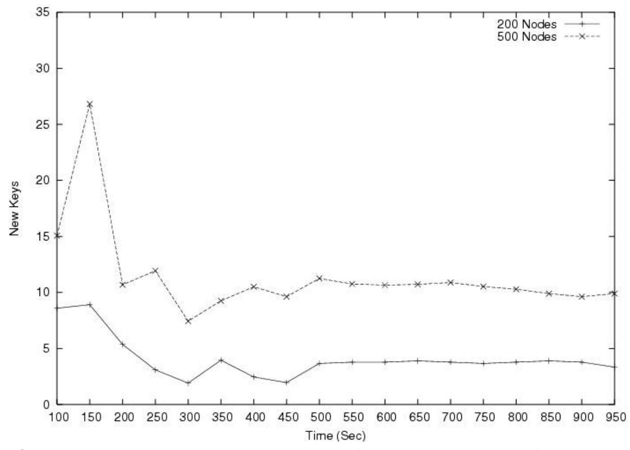

Figure 7: The average number of new keys established per node per 50 sec using PNA with 500 sec key lifetime.

expiration is considered, leads to an increased percentage of operations being dedicated to the keying overhead.

As expected, PNA has significantly higher overhead than RK because as nodes move they establish keys even if no data is ultimately sent. Such an extremely high overhead is the result of sending such a small amount of data between nodes.

Through simulation, it was discovered that new neighbors were typically being added individually as soon as they were detected. In so doing, the number of times the full key establishment protocol had to be executed typically matched the number of nodes added over a given time period. The benefit for such an egregious use of energy, conversely, is decreased routing latency. Low-intensity, latency critical applications such as reporting the presence or identity of a newly acquired target in a test-bed can be expected to demonstrate such characteristics. If, as is the case in this example, the nature of the communication is critical, the steep cost of overhead

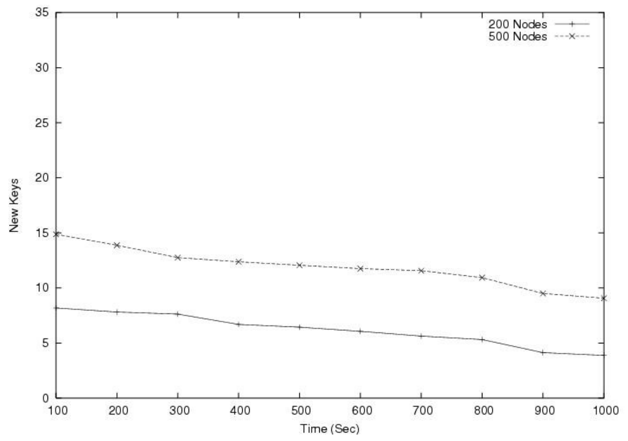

Figure 8: The average number of new keys established per node with RK with transmissions every 100 seconds.

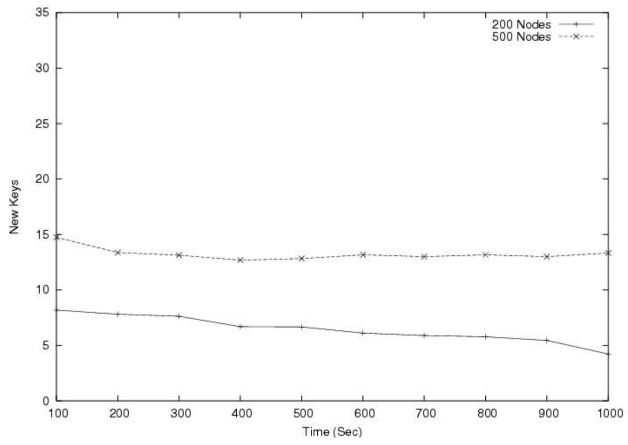

Figure 9: The average number of new keys established per node with RK with transmissions every 100 seconds and a key lifetime of 500 seconds.

is more acceptable; however, non-critical communication with low intensity has been demonstrated as fairly expensive. Should the nature of routes between nodes become more stable, however, the overhead decreases significantly. Because the operating expense of the network appears to be heavily dependant upon these factors, a comparison of traffic intensity to overhead is shown above in Figure 3.

The relationship between traffic intensity and overhead, as demonstrated in Figure 3, exhibits an important attribute of PNA keying. The correspondence between these two elements is linear, thereby solidifying the idea that overhead is not a function of traffic intensity but instead of mobility. As the amount of mobility increases during a given period, so too does the overhead. This relationship is intuitive as the very nature of PNA requires for the establishment of all nodes with which contact is made.

Because the RK approach establishes keys only at transmission time, it incurs overhead at almost a per RREQ rate. This data alone may be enough to be the deciding factor when selecting a keying strategy for a network of mobile sensors. Should the mobility of individual nodes be high and the number of destinations for and frequency of data delivery be low, the RK approach may be more beneficial; however, should mobility be low and the number RREQs become large, an implementation of PNA could prove more advantageous to the operations of the network. Before any judgment is made, it is necessary to consider the consequences of these approaches on packet loss.

Figures 4 and 5 depict the effects of keying on the packet loss ratio. In the first case, static networks, none of the simulations exhibited any dropped UDP packets. This result is not totally unexpected as the frequency of keying in both the PNA and RK cases dies off quickly as neighbors are unchanging. For the second case in which 200 nodes are present in the simulation, we see that the drop rates for the RK scheme exceed that of PNA. There are two main reasons for these losses. The first, and potentially more crucial, is the result of routes being broken due to the latency associated with keying RK keying. The route established by the RREQ is often no longer valid by the time data transmission occurs. The second and more obvious is collisions. While the PNA case was almost constantly attempting to establish secure relationships with new neighbors, the effects of high medium access were effectively amortized as the keying traffic at the time of data transmission was less than in the RK scenario. Similar trending continues when the scenario contains 500 nodes as is seen in Figure 5. The expiration of keys has very little effect on this number as pair-wise connections are maintained through a direct exchange between two nodes should the key expire during the passing of data.

If keys must be renewed, the performance of both PNA and RK will differ. As discussed in Section III, PNA nodes will proactively maintain keys with their current neighbors, whereas RK will only maintain keys with active neighbors.

The effect of key expirations is shown in Figures 6-9. Figure 6 shows the number of new keys being established in each node every 50 seconds when using PNA with an infinite key lifetime. Figure 7 shows the same measurement when the key lifetime is 500 seconds. We observe that when an infinite lifetime is used the number of keys established continues to decrease over time, while with finite key lifetimes steady-state is reached.

RK with and without elapsing keys is demonstrated in Figures 8 and 9, respectively. In both simulations, a message was sent from a source to a destination node once every 100 seconds. As is shown in Figure 7, the speed at which the RK method approaches steady-state keying is much slower than is demonstrated for PNA. Further comparison demonstrates that both node density scenarios result in higher values for the average number of new secure relationships established per epoch.

RK avoids the unnecessary key-exchanges previously discussed in conjunction with PNA; however, this approach requires an additional period of time to reach a steady state in which key establishment is less frequent. While the number of keys established per broadcast closely approaches the average value attained PNA, RK is not quite able to reduce this number to that of its competitor during the 1,000 second simulation. When security associations are allowed to expire as is demonstrated in Figure 8, a node must establish a steady state average of 7.2 (or approximately the average number of neighbors for a network with 200 nodes) keys per broadcast in order to communicate securely.

C. Group Mobility

To study group mobility, five evenly sized groups of nodes were created from 200 nodes around the same number of randomly placed "leader nodes". Publications such as [16] create a similar director node which acts as a cluster head capable of performing the duties of subnetting for its group members. The groups are then given random locations throughout the test-bed and moved in a fashion lacking any rigid formation outside of group membership. Unlike the individual mobility case previously discussed, nodes traveled at between four and five m/s in order to maintain membership in the group. Additionally, during the course of the simulation it is possible that groups become temporarily isolated as should be expected in any system with mobile nodes. Such patterns of movement were made in order to simulate the operations of mobile sensors used in the domain of object tracking or disaster related search and recovery.

Unlike in the previous cases in which all nodes move independently of each other, the group relation of the nodes reduces the ability to generalize about all

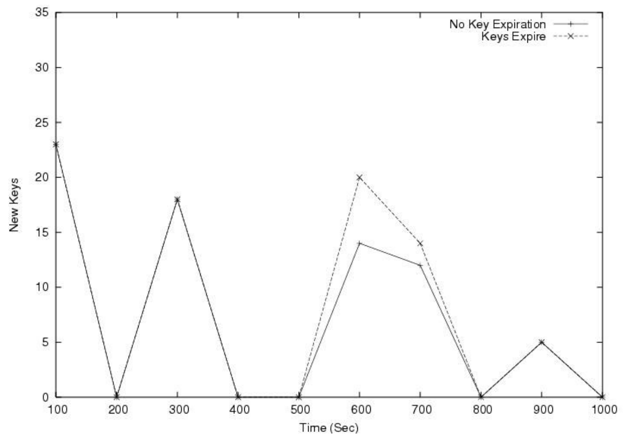

Figure 10: Proactive keying scheme applied to group mobility.

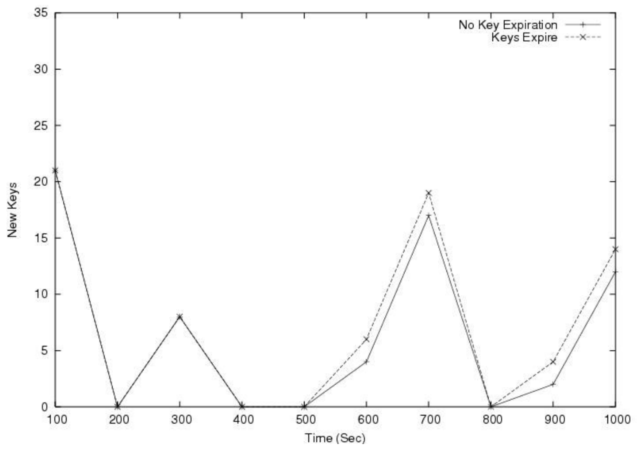

Figure 11: Reactive keying scheme applied to group mobility.

nodes. Whereas it is certainly reasonable and demonstrable to state that each node in the independent model had established a comparable number of secure relationships by a certain time within some margin of error, grouping nodes such that a large cluster may be out of contact with the rest of the network for a time period eliminates our ability to make general statements about trends at given time periods. Therefore, the data presented in Figures 10 and 11 shows a representative time period to illustrate behavior. While the data in both of these figures represents the two keying schemes for a single scenario, additional experiments were run and similar trends were observed.

Figure 10 shows the average number of new keys established per node every 50 seconds when using PNA with infinite and finite key lifetimes. Nodes within a group always maintain keys with their neighbors, so new keys are established as groups come in contact with each other. Depending on the amount of overlap, a varying number of new keys are established. For example, at 250 seconds, two groups come into contact. Over the next 150 seconds nodes establish new keys. At the 400 second mark the groups are apart, so no new keys are established.

At 550 seconds, two groups come into contact again. During this time we can see the impact of having finite key lifetimes. In this case, some members of the groups have previously been in contact, some in the case of infinite key lifetime, fewer new keys must be established. However, with a 500 second key lifetime, a number of the previously established keys have expired, and must therefore be reestablished.

Figure 11 demonstrates a similar snapshot when RK is used with a similar result.

An advantage gained simply by having nodes organized in such a fashion is the reduction of the majority of unnecessary key establishments. In the PNA case especially, the fundamental modification of mobility models such that the frequency of neighbor changes is reduced greatly impacts the necessary regularity of the key protocol's execution. Additionally, because the number of new nodes being detected at any moment is much more likely to be greater than one due to clustering, the number of requisite iterations of the keying protocol itself are also reduced. Lastly, because the time during which two groups may be in contact could be limited by the velocities by which they are moving, the use of PNA allows for keys to be established immediately so as to increase the probability that a data packet needing to be exchanged between the two groups is actually able to be sent. Due to the latency inherent to setting up secure relationships, the RK scheme is more likely to be unable to deliver a packet between groups as one or more of the groups may be moving away from the other at such a velocity that the relationship between the two clusters is overwhelmingly transient.

Both approaches to keying perform in similar fashions in the group mobility setting. Because groups are often out of contact with each other, the ability to establish keys for secure communications between members of different groups is limited. As the number of times that groups move within communication range of each other decreases, the characteristics of PNA begin to degrade to the RK approach, especially if keys expire. If groups are not to be highly mobile, as may be the case of sensors incorporated into the products on pallets in a warehouse waiting for distribution, the mobility model itself becomes equivalent to the static case discussed earlier.

VI. CONCLUSION

Secure routing is an extremely important element of safeguarding sensor network; however, the majority of previous works in this area neglect to specifically address the difficult task of key management. Although a number of papers have suggested the use of such a keying scheme, none have thus far specified the protocol necessary to perform the necessary indirect secure relationships inherent to this approach.

Furthermore, none have applied it to an application such as secure routing and observed the issues inherent to its implementation.

We have shown how the use of a probabilistic key management approach affects latency, overhead and the packet loss ratio in environments where static, individually mobile and mobile groups of sensors are present. If keying occurs proactively such that all new nodes are automatically keyed regardless of the transience of the pair relationship the latency of data delivery is greatly reduced, but at a great increase in overhead. With reactive keying, the number of spuriously established secure relationships that are established is minimized, but only at the cost of tremendously high latency. This latency is often the reason for packet loss when the mobility of individual nodes becomes high. From these results, we have demonstrated that the overhead associated with these methods of key management is a function of the mobility of the nodes and not of the traffic intensity.

If an emergency were to arise, it may take the reactive approach too much time to establish a route for its data to be useful. However, because sensor nodes are absolutely constrained by power restrictions, the reactive scheme may be the only practical scheme that can be implemented in a network that is meant to be long lived. We therefore recommend that a hybrid method of key management be investigated such that the positive attributes of the two above schemes can be combined to make secure routing in sensor networks as robust as possible.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "*A Survey on Sensor Networks*," IEEE Communications Magazine, August 2002.

[2] Atmega128L Microcontroller Features, available at http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf

[3] B. Awerbach, et al., "*An On-Demand Secure Routing Protocol Resilient to Byzantine Failures*," Proc. ACM WiSe, 2002.

[4] S. Capkun, et al., "*BISS: Building Secure Routing out of an Incomplete Set of Security Associations*," Proc. ACM WiSe, 2003.

[5] H. Chan, A. Perrig, D. Song, "*Random Key Predistribution Schemes for Sensor Networks*," Proc. IEEE S&P, 2003.

[6] Crossbow, Wireless Sensor Networks, available at http://www.xbow.com/Products/Wireless_Sensor_Networks.htm

[7] W. Du, J. Deng, Y.S. Han, S. Chen, P.K. Varshney, "*A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge*," Proc. IEEE Infocom, 2004.

[8] W. Du, J. Deng, S. Han, P.K. Varshney, "*Establishing Pairwise Keys in Distributed Sensor Networks*," Proc. ACM CCS-03.

[9] L. Eschenauer and V. Gligor. "*A key management scheme for distributed sensor networks*," Proc ACM CCS, 2002

[10] J. Hubaux, et al., "The quest for security in mobile ad hoc networks," Proc. ACM MobiHOC, 2001.

[11] Y.-C. Hu, et al., "*A Survey of Secure Wireless Ad Hoc Routing*," IEEE S&P, 2004.

[12] Y.-C. Hu, et al., "*Ariadne: A secure on-demand routing protocol for ad hoc networks*," ACM MobiCom, 2002.

[13] C Intanagonwiwat, R Govindan, D Estrin, "*Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks*," Proc MobiCom, 2000.

[14] D. Liu, P. Neng, "*Establishing Pairwise Keys in Distributed Sensor Networks*," Proc. ACM CCS'03.

[15] P. Papadimitratos and Z.J. Haas, "*Secure Routing for Mobile Ad Hoc Networks*," Proc. SCS CNDS, 2002.

[16] G. Pei and M Gerla, "*A Wireless Hierarchical Routing Protocol with Group Mobility*," Wireless Communications and Networking Conference, 1999.

[17] A. Perrig, et al., "*SPINS: Security Protocols for Sensor Networks*," ACM Wireless Networking, September 2002

[18] A. Perrig, R. Canetti, D. Tygar, and D. Song, "*The TESLA Broadcast Authentication Protocol*," *RSA CryptoBytes*, 5(2):2--13, 2002.

[19] K Sanzigiri, et al, "*A Secure routing Protocol for Ad Hoc Networks*," Proc. IEEE ICNP 2002.

[20] P. Traynor, H Choi, T La Porta and G Cao, *Establishing Pair-Wise Keys in Heterogeneous Sensor Networks*, Network and Security Center Technical Report TR-NAS-0001-2004,

Department of Computer Science and Engineering, Penn State University, November 2004.

[21] S. Yi, et al., “*Security-Aware Ad Hoc Routing for Wireless Networks*,” Proc. ACM MobiHOC, 2001.

[22] C. Yin, et al., “*Secure Routing for Large-Scale Sensor Networks*,” Proc International Conference on Communication Technology, 2003.

[23] J. Yoon, et al, “Random Waypoint Considered Harmful,” Proc IEEE Infocom, 2003

[24] M. Zapata, “*Secure ad hoc on-demand distance vector routing*,” ACM SIGMOBILE Mobile Computing and Communications Review, 2002.

[25] S. Zhu, et al., “*Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach*," Proc. IEEE ICNP 2003.