Energy-Efficient Hybrid Key Management Protocol for Wireless Sensor Networks

Tim Landstra, Sarangapani Jagannathan, and Maciej Zawodniok
(Corresponding author: Maciej Zawodniok)

Missouri University of Science and Technology
301 W. 16th Street, Rolla, MO 65409, USA
(Email: mjzx9c@mst.edu)
(Received Dec. 6, 2006; revised and accepted June 12, 2007)

Abstract

In this paper, an energy-efficient hybrid key management (EHKM) protocol is proposed in which the heterogeneous security requirements of a wireless sensor network are considered to provide differing levels of security with minimum communication overhead. Additionally, it allows the dynamic creation of high security subnetworks within the wireless sensor network and provides subnetworks with a mechanism for dynamically creating a secure key using a novel and dynamic group key management protocol. The proposed energy-efficient protocol utilizes a combination of pre-deployed group keys and initial trustworthiness of nodes to create a level of trust between neighbors in the network. This trust is later used to allow secure communication between neighbors when creating a dynamic, high security subnetwork within the sensor network. This static and dynamic key management combination creates a hybrid key management protocol. Analysis of the proposed protocol with network and cluster sizes and different node failures is performed using the Ns2 network simulator. Additionally, the protocol is compared to other protocols, for example LEAP and a recent dynamic group key management protocol. Additionally, the security of the proposed protocol is analyzed against various attacks by an adversary. Also the overhead due to communication and computation is investigated.

Keywords: Key management, sensor network security, wireless security

1 Introduction

Wireless sensor networks (WSN's) have recently come into the forefront of research due to their possible uses in military and disaster relief cases. Additionally, the medical and commercial fields have found potential uses for sensor networks [1, 3]. A WSN is a highly constrained type of network, comprised of sensor nodes with limited capabilities and larger gateway nodes, referred to as cluster heads, with more capabilities. In a WSN, the toughest constraints include the limited available energy and memory. Thus, lightweight, energy-efficient security protocols are necessary for these networks. The communication medium used for sensor networks is an additional factor that must be considered when designing security protocols. Sensor networks use a radio frequency channel which is susceptible to eavesdropping. Additionally, individual nodes in the sensor network may be compromised by an adversary and the network must be able to operate in the presence of such an attack.

Typically, there are two types of key management protocols available to a network designer. Keys of an either symmetric or asymmetric nature may be used in a key management protocol. Asymmetric cryptography is typically considered to be too computationally intensive for use in sensor networks, so symmetric keys are traditionally used. In general, a set of pre-deployed keys have to be stored at each node in WSN in order to initialize secure communication. Next, a dynamic key management scheme is needed to create new and revoke old or compromised keys. Rigorous work has been published on random pre-deployment of keys [2, 5, 6, 8, 10, 19] where a limited number of pair-wise or partial keys are randomly assigned to nodes before deployment. In general, the number of necessary pre-deployed keys or partial keys increases with network size [2, 5, 6, 8, 10, 11]. In Matt et al. [1], it is shown that as a network's size increases, the number of keys that must be deployed with the network increases exponentially. A prior or explicit knowledge about deployment as used in [5, 8] and [10] can only reduce the number of required pre-deployed keys, but it still will increase with network size. Other papers [11, 15, 18] reduce the number of required pre-deployed keys without need for assumption of a prior knowledge about deployment. For example, [15] investigate the use of pre-deployed key rings to minimize the number of keys that must be deployed with a node in order for it to be likely that the node is in communication range with another node with a matching key. In [11], the polynomial pool and hypercube-based

pre-distribution schemes can also reduce number of pre-deployed keys. In contrast, the proposed scheme requires only 3 keys to be initially stored by each node regardless of the network size and deployment pattern. Consequently, the scheme is a fully scalable solution which simplifies pre-deployment preparations.

Also, the use of dynamic key management techniques as a mechanism to deliver security in WSN's was investigated in [4, 7, 12, 13, 20], and in case of in ad hoc networks in [9]. Dynamic keys are preferred over pre-deployed keys because they minimize the number of keys that must be stored by nodes. Additionally, the key can be changed frequently, decreasing the ability for adversaries to determine the network's key. This secure and dynamic group key management (DGKM) protocol utilizes a modified tree based group Diffie-Hellmann key management scheme [7]. This protocol allows clusters in a sensor network to dynamically create a key. The nodes in the cluster create partial keys from the leaves of the group up to the root. Nodes use the partial keys of their children as inputs to the function $\alpha^k \mod p$ where p is a prime number, k is the result of an *xor*'ing of two of the node's children's partial keys and α is a primitive root of p. The cluster head aggregates the partial keys of all the nodes and creates a group key which is then broadcasted to the group.

The majority of previous literature considers the needs of security in a WSN to be homogenous in nature. In practice, this may not be true as many nodes in a sensor network may be in an idle state or transmitting information of little importance most of the time. These nodes do not need to operate at a high security level that consumes large amounts of energy. This paper introduces a novel security protocol for sensor networks that utilizes the heterogeneous nature of the security requirements of WSN's to realize distinct security and energy levels within the network. EHKM allows the majority of a network to operate in a low security mode with static keys that conserves energy, while dynamically creating keys for high security subnetworks. This dynamic key creation protocol uses a subtree energy average function to determine the subtree or subtrees to be used to create the dynamic partial keys. The function accepts the number of partial keys the subnetwork wishes to create and then selects a number of nodes to create partial keys that is equal to or slightly greater than the number of desired partial keys. This subtree energy average function conserves energy over the DGKM protocol since the number of nodes in the subnetwork may be much greater than the number of partial keys desired for the protocol.

Furthermore, to enable secure communication between any pair of neighbor nodes a session key has to be established. Typically, a shared key is used since low complexity and energy requirements of the shared-key encryption schemes. Such a pair-wise key can be either generated from pre-deployed partial keys [2, 5], or found among the pre-deployed keys [8, 10, 11, 19], or established through a third party who shares the pair-wise keys with each of the nodes in the pair [2, 5, 8, 10, 11, 19]. In contrast, the proposed scheme dynamically creates pair-wise keys between neighbor nodes during short initial post-deployment phase. The scheme reduced energy consumption by employing simple and basic cryptography mechanisms. When a high security subnetwork is created, the stronger key is setup at the expense of higher energy consumption. However, the overall energy consumption is reduced when compared with homogenous schemes which always have to use the high security mechanisms. In contrast, the proposed scheme employs the energy-expensive high security methods only when needed.

The proposed protocol utilizes a hybrid pre-deployed/dynamic key management protocol to realize two distinct security and energy levels within the network. All nodes are pre-deployed with a secret key that is common to all nodes in the network. The nodes use this key for broadcast messages that have low security requirements. Additionally, upon deployment, nodes use a second common pre-deployed key to create an individual key and trust between neighbors. Since it is assumed that nodes are safe from capture and compromise for a period of time after deployment [20], nodes able to communicate with the second secret key are assumed to be a legitimate part of the network and a trust and key may be established with them.

Upon creation of a subnetwork with high security needs, it is assumed there are several cluster heads within communication range of each other. The cluster heads compute the average energy per node in their respective cluster and broadcast the value to the other cluster heads. The cluster head with the highest average energy left in its nodes is elected the head cluster head (HCH). Further cluster heads are also selected until enough the number of nodes in all the chosen clusters is greater than the number of partial keys desired for secure communication. Once enough nodes are selected, nodes in the chosen clusters participate in the DGKM protocol. Nodes use the individual keys established upon initialization of the network to communicate this information, keeping it separate from the common secret key of the entire network. Once a subnetwork key is created, it is utilized for the lifetime of the subnetwork until the subnetwork is disbanded and the nodes return to the lower security level. As a result, the protocol relaxes the need for homogenous security. We observe that often a sensed event will span over more than one of the clusters present in a typical sensor network. Thus, all the nodes in each cluster that is affected must come out of idle state and several keys must be used to communicate the event back to the base station. This observation was addressed recently in [16] from a routing perspective and our protocol further improves security and energy efficiency.

In this paper, EHKM is analyzed against other protocols in literature in terms of complexity of the key management algorithm and communication and storage overhead. Simulations are performed in Ns2, comparing the energy consumed in the simulation environment to other

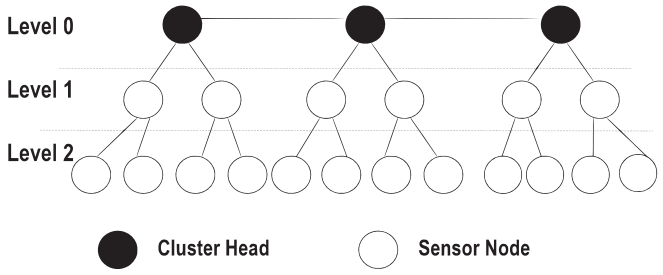

Figure 1: Example of sensor subnetwork architecture

protocols in a fully functioning network as well as in a network with node failures. It is found that the protocol consumes less energy, uses less messages and is more scalable than LEAP [20] and the Dynamic Key Management Protocol [13] and provides a similar level of security.

The main contributions of this paper include: using the subtree average function to decide which nodes to use to generate the partial keys in an energy-efficient manner; and combination of a static and dynamic key management protocol to create a hybrid protocol. The rest of this paper is laid out as follows. Section 2 discusses the sensor network structure this protocol will be utilized in and Section 3 then covers the operations of the protocol. Section 4 shows the energy and complexity analysis of the protocol and Section 5 shows simulation results. Section 6 concludes the paper and covers further works.

2 Sensor Network Architecture

WSNs are often deployed remotely and network designers have no control over the location or placement of the sensors in either a rough terrain or relation to other sensors around them. This paper considers such a random node deployment over a rectangular topography of varying size. Once the nodes have been deployed, it is assumed that there is a period of time T_{min} where the nodes are not being compromised by an enemy and can safely exchange keys [20]. Before the time T_{min}, the nodes are responsible for broadcasting their individual key to their 1-hop neighbors. These individual keys can be created by generating a random number. When a sensing event occurs within the network, it is assumed that the nodes around the event form into a subnetwork with a hierarchy. The energy-efficient self-organizing protocol (SOS) discussed in [16] is employed in this application to create a subnetwork around an event. The subnetwork includes a set of nodes that are designated as Cluster Head's (CH's) that are above the other nodes in the subnetwork in the hierarchy.

The SOS protocol assumes the network is monitoring localized, random events. Nodes sensing an event will asses their proximity to the event epicenter by measuring the strength of the sensed parameter. If the measured signal strength is greater than a given threshold, then such nodes will group themselves into a subnetwork. Once the subnetwork has been established, the nodes exchange their available energy. Next, a number of cluster heads is chosen based on energy remaining, proximity to the event, and size of the subnetwork. Then, nodes join the closest cluster head. In case of nodes being more than 1 hop away from CH, the sensors can form a spanning tree with the CH as a root. Figure 1 shows an example clustering of such a network.

Nodes are expected to know the number of nodes within its subtree and within the same level as itself. Let denote n_{ij} be the number of nodes that are part of subtree i on level j. For example, in Figure 1, $n_{1j} = 2$ and $n_{2j} = 4$ for all $j = 0, 1, 2$, for every subtree, there are 2 nodes on the first level and 4 nodes on the second level. However, a particular network topology cannot be assumed beforehand since a particular node may have different number of children nodes. Additionally, cluster heads are expected to know the remaining energy of all the nodes that are members of their cluster. These assumptions will all be fulfilled since the SOS protocol [16] is used.

3 Protocol Details

This protocol uses two separate key management schemes; one for group-wide and individual keys and another for subnetwork key management. The group-wide key is used for non-critical broadcast messages between nodes. The individual keys are used for secure communication between nodes creating a subnetwork and setting up a subnetwork key. The second key management scheme is creating and distributing the keys for the dynamically created subnetworks. Securely distributing the keys for the subnetworks created by events within the sensor network is a non-trivial problem since the subnetworks may contain any arbitrary set of neighboring nodes. These nodes all must have a mechanism to securely communicate with each other to distribute the subnetwork key to all the subnetwork members.

3.1 Group-Wide and Individual Keys

The protocol begins with deployment of the network. Before the network is deployed, three values must be stored in each node. Two of these values will be common to each node in the network. The first, K_1, is a group-wide key that will be used for group-wide communication between nodes not involved in a subnetwork. The second value, K_2, will be used for exchanging pair-wise keys between neighboring nodes and will be erased after time T_{min}, similar to the LEAP protocol [20]. The third value, K_3, is unique to each node and is stored on both the node and the base station. This key can be used for private messages between a node and the base station.

After deployment, nodes begin by transmitting a HELLO message before time T_{min} has elapsed. The HELLO message will be structured in the following way:

$N_i \rightarrow S_i : N_{ID}, E_{K_2}(nonce), MAC(K_2, N_{ID} \| nonce)$
where S_i is a set of all one-hop neighbours, N_i indicates the transmitting node, N_{ID} is its address, $E_{K_2}(nonce)$ is the *nonce* encoded using key K_2; and $MAC(.)$ is the message authentication code. The random nonce included in the HELLO message serves as the nodes secret key for any messages it sends during the network's lifetime. A node which receives the HELLO message decrypts the nonce and verifies the message using the MAC. After a successful verification, the node stores the ID, *nonce* pair. Thus, it can decrypt any non-broadcast message sent by the neighbor by using the nonce corresponding to the source's ID.

Once T_{min} has elapsed, the nodes are responsible for deleting K_2 from their memories such that it cannot be retrieved by enemy once a node is captured. Afterwards, subnetworks can be dynamically created in response to events. Next, the corresponding subnetwork key management protocol and algorithm are presented.

3.2 Subnetwork Key Management Protocol

The subnetwork section of the key management protocol begins with the creation of a subnetwork within the WSN. The cluster heads initiate a procedure by finding the average energy remaining for all the nodes in their cluster $E(i) = \frac{\sum_j E_{ij}}{n_i}$, where $E(i)$ is the average energy per node in cluster i which CH_i is responsible for calculating, E_{ij} is the energy left in the j^{th} node of the ith cluster and n_i is the number of nodes in cluster i. The cluster heads then send this value and the number of nodes in their clusters to the other cluster heads. Then the subnetwork chooses a cluster head to be the Head Cluster Head (HCH). A cluster head with the highest average remaining energy is selected as HCH:

$$HCH = CH\left(max\left\{E(i)\right\}\right)$$

The network has a predetermined number of partial keys that are sufficient to create a subnetwork key. The minimum number of partial keys, m, can be any value such that m partial keys of length l will create a subnetwork key of sufficient length $(m * l)$ to be secure for the length of time the subnetwork is expected to be active. This number has been suggested to be equal to 15 in past literature [13]. Nodes in the HCH cluster form an initial pool of partial keys used to create the subnetwork key. Next, the other cluster heads check to see if the number of nodes in the HCHs cluster, n_{HCH}, is sufficient to generate the subnetwork key (that is $n_{HCH} \geq m$). Otherwise, nodes of the CH with the second highest average remaining energy are added to the pool. This method continues until the number of nodes in the pooled clusters is greater than m.

Once a sufficient number of cluster heads have been chosen to generate the necessary m partial keys, the cluster heads begin the key generation algorithm by broadcasting a *start_algorithm* message to their nodes with the cluster ID and the depth into the cluster that the message should go before nodes begin creating partial keys. The depth field in the messages allows the algorithm to restrict partial key creation to the upper levels of the hierarchy to conserve energy. For all the clusters chosen other than the last one, the depth field is set to -1. This means that all nodes in the cluster must generate a partial key. For the last cluster that was chosen, the depth field is set to $(m \cdot sum(n_{other\_clusters}))$. That is, the number of partial keys desired, minus the number of nodes in all the other selected clusters. As a result, exactly the required m partial keys are used during the subnetwork key generation, thus conserving energy during calculations.

When a node receives a *start_algorithm* message from its cluster head, it checks the depth field to see if it is equal to -1. If it equals -1, then the node rebroadcasts the message to all its children nodes. If the node is a leaf node, then it creates its partial key and sends it to HCH through its parent. In case the depth field is not equal to -1, the node subtracts the number of nodes in its level and cluster. Consequently, it will determine if more nodes are required to participate in the key generation. If the depth field is still greater than 0, then the node rebroadcasts the message. Otherwise, the node acts as a leaf node and randomly creates its partial key and sends it to HCH through its parent.

Once the HCH receives the required m partial keys from the subnetwork, the HCH constructs the subnetwork key and broadcasts it to the subnetwork. This subnetwork key is encrypted with the individual keys as it is transmitted throughout the subnetwork.

3.3 Dynamic Cluster Key Management Algorithm

Once the network determines the clusters that will be creating the partial keys, a modified version of the dynamic group key management protocol proposed in [13] is used to generate the subnetwork keys. In this protocol, the leaves begin the protocol by randomly generating a partial key. This key is then passed to the parent of the leaf node. Once the parent receives the partial keys from two of its children, it is able to create its own partial key by combining the two keys using a function f(partial key child 1, partial key child 2). This function is represented as

$$f(k1, k2) = a^{k1 \oplus k2} \bmod p,$$

where p is a prime number, α is a primitive root of p, and k_1 and k_2 are the children's partial keys $(k1, k2 < p)$. The parent then passes its partial key to its parent and passes the partial keys of all its children to the cluster head. The cluster head collects all the partial keys and combines them to form the cluster key. It then broadcasts the key to the cluster.

By contrast, the proposed protocol introduces several modifications in order to accommodate different valid

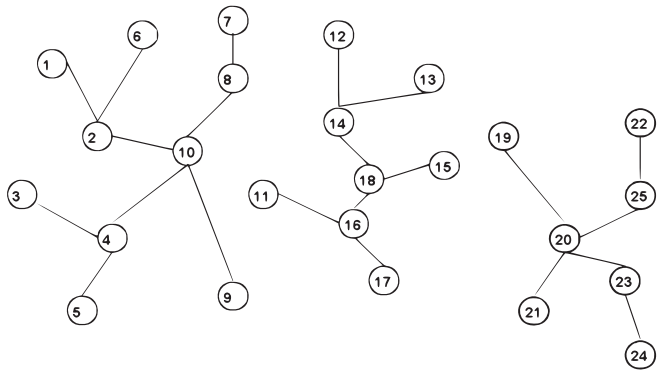

Figure 2: Example subnetwork

node topologies. For instance, the proposed protocol no longer assumes that all nodes are either leaves or have more than one child. If a node only has one child, it waits to receive its child's partial key then generates a random number as the second partial key for the algorithm. Additionally, instead of generating a key for only a cluster, the algorithm is expanded to generate a key for an entire subnetwork.

3.4 Example

This section gives an example of how the subnetwork would choose the clusters to participate in the partial key creation algorithm. In this example, a network has been deployed and a subnetwork with multiple clusters has been created. Consider a subnetwork, show in Figure 2, which contains three clusters: the leftmost "Cluster 1", the middle "Cluster 2" and the rightmost "Cluster 3". The numbers of nodes in each cluster is as follows: $n_1 = 10, n_2 = 8, n_3 = 7$.

It is assumed that the number of minimum partial keys, m, is equal to 12 in this example. The protocol will progress in the following way:

1) Cluster heads 10, 18, 25 find the total energy left in their respective clusters and divide by the number of nodes in their cluster. Let's assume that the average available energy values are computed as: $E(1) = 0.9[J]$, $E(2) = 0.95[J]$, $E(3) = 0.92[J]$, where $E(1)$, $E(2)$ and $E(3)$ are the average energy levels at Clusters 1, 2 and 3 respectively.

2) The cluster heads transmit the $E(i)$ and n_i values to the other two cluster heads. Once all the cluster heads have received these messages, the Cluster 2's cluster head {18} is selected as the HCH since it has the most energy left per node.

3) The clusters subtract n_2 from the value m to find that Cluster 2 does not have enough nodes to produce the needed partial keys. It is found that $m - n_2 = 4$ and that another cluster will need to provide 4 partial keys. The next highest average available energy value is found to be $E(3)$, thus Cluster 3's CH {25} is added to the list of key generation pool of cluster heads.

4) The total number of node in both chosen cluster is greater than m, $(n_2 + n_3 \geq m)$, thus no further cluster head needs to be chosen. Node 18 broadcasts a *start_algorithm* message to its cluster with the cluster ID of 2 and the depth field set to -1. Node 25 broadcasts a *start_algorithm* message to its cluster with a cluster ID of 3 and the depth field set to 4, $(m - n_2)$.

5) The first level of Cluster 2 and Cluster 3 receive their respective messages. The first level of Cluster 2 {14, 15, 16} simply rebroadcast the message they receive since the depth field is set to -1. The nodes on the first level of Cluster 3 {20, 22} subtract the value $n_{1,3}$ (number of nodes on the first level of Cluster 3) from the depth field and rebroadcast the message with the new depth field. In this case, the new depth field will be equal to 2 $(4 - 2)$.

6) The second level of Clusters 2 and 3 receive the *start_algorithm* messages. In the case of Cluster 2, all these nodes are leaf nodes and they begin the partial key algorithm by creating their partial keys and transmitting them to their parents. In the case of Cluster 3, not all the nodes are leaf nodes, so the second level subtracts the value n_2, 3 from the depth field. This new value is equal to $2 - 3 = -1$. Hence, no retransmission of the *start_algorithm* message is necessary. All the nodes on Level 2 of Cluster 3 begin the partial key algorithm by creating their own partial keys and transmitting them up the subtree.

7) As node 25 receives partial keys from its cluster, it passes them onto the HCH, Node 18.

8) Finally, the HCH (Node 18) will receive all the partial keys that have been generated by the algorithm. In this example, Node 18 would receive a total of 13 partial keys (assuming no partial key packets were dropped). This is greater than m, but less than the number of partial keys that would be received if the entire subnetwork were involved in the partial key creation algorithm. From these 13 partial keys, the HCH would create the subnetwork key and broadcast it to the members of the subnetwork using the individual, secret keys set up at the initialization of the network.

The following section contains a C-like pseudo-code of the proposed protocol.

3.5 Pseudo-code

```
I. Initialization of network
   1. For each node in network
      Broadcast Hello: {ID || EK2(ID || nonce)}
   2. If hear a Hello
```

```
      Verify ID, store ID / nonce pair
II. After creation of a subnetwork
   1. Set cluster_heads = {Each cluster head}
   2. For each cluster head
      set avg_energy(cluster) =
            average(energy_in_nodes_in_cluster)
      set ncluster = number of nodes in cluster
   3. Set HCH = {Cluster head:
                 max(avg_energy(cluster_heads)}
   4. Set chosen_CH = {HCH}
   5. cluster_heads = {cluster_heads} - {HCH}
   6. Set m_temp = m - n_{HCH}
   7. while m_temp >= 0
       CH_temp = {Cluster head:
                 max(avg_energy(cluster_heads)}
       chosen_CH = {chosen_CH} - {CH_temp}
       m_temp = m_temp - |CH_temp|
   8. Set n_chosen = 0
   9. Set depth = -1
  10. while |chosen_CH| > 1
       CH_temp = first_element(chosen_CH)
       CH_temp broadcasts start_algorithm
                          {Cluster_ID || depth}
       n_chosen = n_chosen + |CH_temp|
       chosen_CH = {chosen_CH} - {CH_temp}
  11. depth = m - n_chosen
  12. CH_temp = only_element(chosen_CH)
  13. CH_temp broadcasts start_algorithm
                          {Cluster_ID || depth}
III. After hearing a start_algorithm message
   1. If node is a leaf
       Start_algorithm(.); return
   2. Else
   3. depth = received_depth - n_ij
         (where n_ij is number of nodes in
                       cluster i on level j)
   4. If depth <= 0
       Start_algorithm(.); return
   5. Else
   6. Broadcast start_algorithm
                          {Cluster_ID || depth}
```

4 Performance Analysis

In this section, a performance analysis of EHKM is presented. This includes an analysis of the complexity of the algorithm and the storage costs imposed upon memory. Also, an analysis of the number of messages that are sent is included. Finally, simulation results from Ns2 with a variety of network configurations and a varying number of node failures are analyzed.

4.1 Complexity Analysis

Table 1 presents the results of the complexity analysis on the protocol and compares it to several existing key management schemes. The message cost incurred and complexity of the proposed algorithm are less than for the LEAP and the dynamic key management since it is typical to have the number of desired partial keys, m, be much greater than number of all nodes in the network, N, $(n << N)$.

For the Table 1, the degree of the network is defined as the average number of nodes that are within communication range of a given node. It is proportional to the density of nodes in the network and may be a value from 10 to 20 for a reasonably dense network. The setup cost of the protocols is measured in the number of messages that must be sent by the network in order to initialize the key management protocol. For EHKM, each node must send one message to initialize the protocol. This leads to N messages being sent by the network. The LEAP protocol has several rounds of key exchange between nodes and their neighbors individually. This leads to a number of message proportional to N as well as d.

The number of messages that need to be sent by EHKM for the key management of the network is measured from the time a subnetwork forms to the time a subnetwork key is successfully distributed to the nodes of the subnetwork. First the NCH cluster heads must distribute the average energy left in their clusters. This results in NCH messages. Then the *start_algorithm* message is distributed to m nodes who reply, both to their parent and to their cluster head. As a result, $3m$ messages will be sent. Finally, the subnetwork key is broadcasted to the NC nodes of the subnetwork. The time complexity of this stage is $log(m)$ since the $3m$ messages dominates the NC and NCH messages. The LEAP protocol requires several rounds of messages within a localized area and then two rounds of broadcasting. For the dynamic protocol, there are three rounds of messages for every cluster in the network, leading to $3N$ messages being sent.

The memory requirement of EHKM is measured in terms of number of bytes used to store the keys of the key management protocol. For the proposed protocol, a node must keep all the individual keys of nodes within communication range of it. This is measured by the density of the network, d. The number of bytes used to store the individual keys is $d * KL$. The node must also store the network-wide key and the key it shares with the base station. Then each node must also store the subnetwork key, which is of length k. The LEAP protocol stores 3 keys for each neighbor, an individual key, a group key and a key chain with L values. All of these keys have length KL.

In comparison, EHKM has a lower storage cost than the LEAP protocol ($d*KL$ versus $3*d*KL$) but will have a comparable storage cost to the dynamic protocol for current network sizes. For a network with the following properties $(d = 20, N = 1000, m = 20, KL = 10, k = 14)$ both the proposed protocol and DGKM protocol require 214 bytes of memory for key storage.

Figure 3 illustrates the storage cost of the LEAP algorithm and the proposed algorithm with varying degree of

Table 1: The general description of the head-end

	Setup		Key Management		Storage
	Messages	Complexity	Messages	Complexity	
Leap	$N(1+5d)$	$O(N)$	$2(d-1)^2/N+2N$	$O(d^2)$	$(3d+2+L)KL$
Dynamic	-	-	$3N$	$O(\log N)$	$m\log(N+k)$
Proposed	N	$O(N)$	$3m+NS+NCH$	$O(\log m)$	$d*KL+k$

d is average degree of the network, N is number of nodes in network, NS is number of nodes in a subnetwork $NS < N$, m is number of desired partial keys, KL is individual key length, k is dynamic subnetwork key length, L is length of key chain; and NCH is a number of cluster heads.

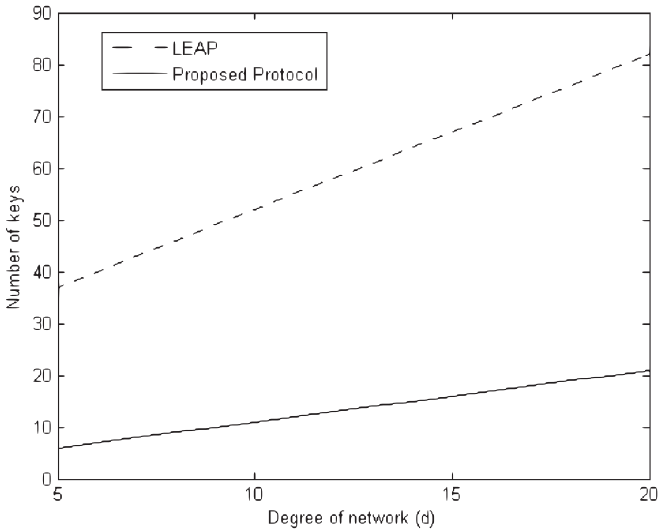

Figure 3: Comparison of storage costs

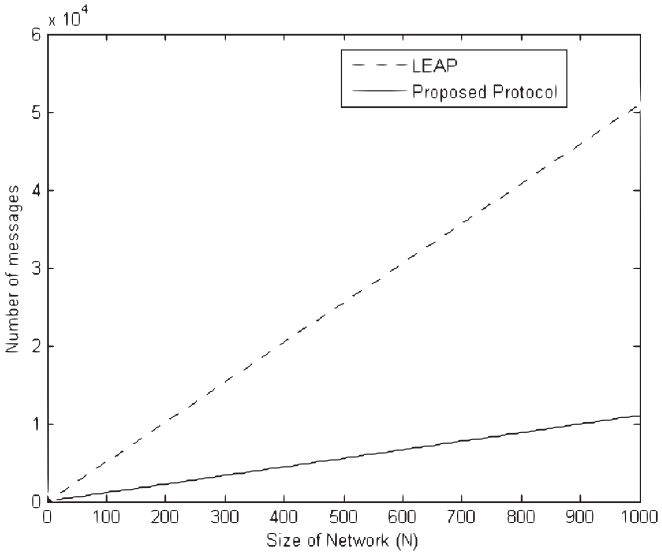

Figure 4: Comparison of number of messages

network. It was assumed that deployment area remains constant while number of nodes increases, thus increasing the density of the network. EHKM needs to store far fewer keys than the LEAP protocol for any degree of network as observed in Figure 3. As network size increases or decreases for any given degree, the number of keys will not change for both protocols. In general, the proposed protocol will need to store fewer keys than LEAP.

Figure 4 shows a comparison of the number of messages needed to bootstrap a deployed network using either the LEAP protocol or EHKM. This includes all the messages sent from the nodes deployment to the end of the key management protocol initialization. The proposed protocol scales better with network size than the LEAP protocol. These results show that EHKM will initialize faster and with less communication cost and storage cost than the LEAP protocol. Comparison to the dynamic key management protocol is shown in the next section with simulation results.

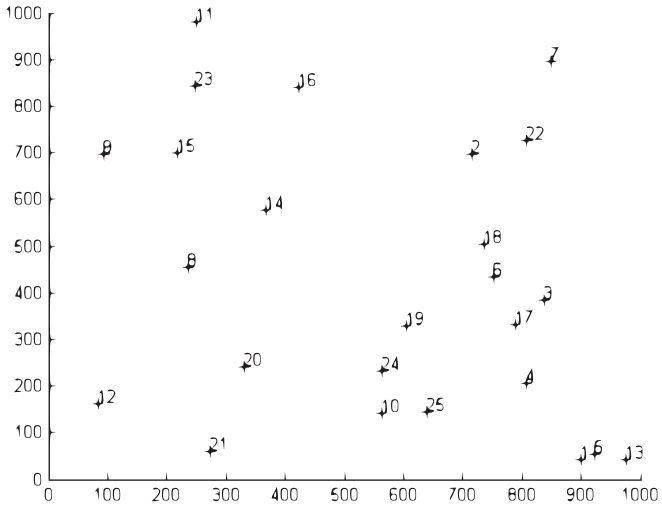

Figure 5: 25 node topology

4.2 Simulation Results

EHKM was simulated in Ns2 and compared to the dynamic key management protocol. The simulation results are shown in the following subsections. For the simu-

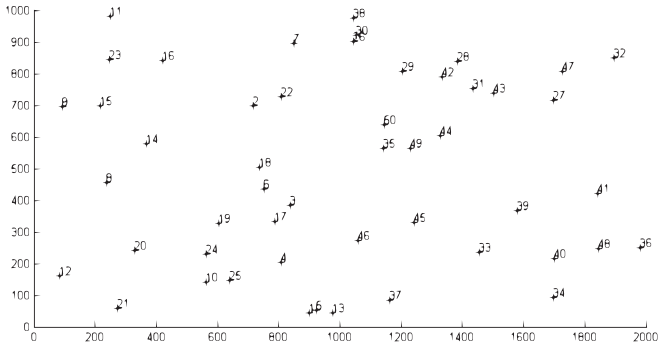

Figure 6: 50 node topology

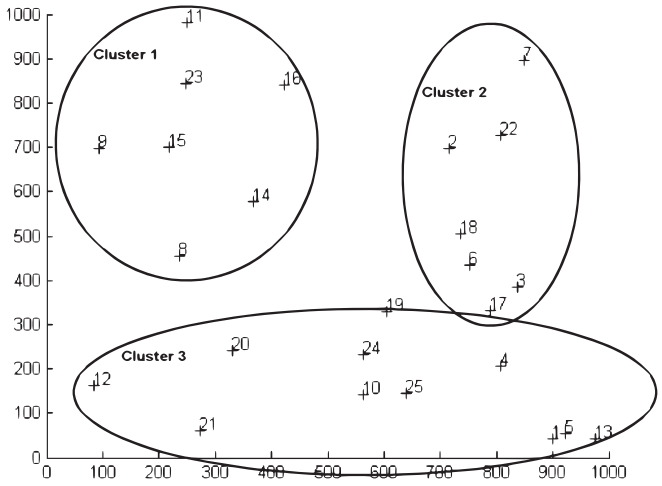

Figure 7: Orientation of network

lations, a 2-ray ground reflection model was used with antennas 1m above the ground. A random distribution of 25, 50 and 100 nodes was simulated on topography of 1000m x 1000m, 2000m x 1000m and 2000m x 2000m respectively, to keep node density at a constant 2.5x10-5 nodes / m2. A subnetwork size of 25 nodes was used and the subnetwork was split into three clusters whose size was based on orientation of nodes in relation to assigned cluster heads. The number of partial keys desired was set to 7 for this simulation and partial key size messages were set to 4 bytes (32 bits). This would result in a 224-bit subnetwork key that can for example be used in 3TDEA algorithm [17]. The routing protocol used was the ad-hoc on-demand distance vector (AODV). The networks were simulated from network deployment until the establishment of the subnetwork's first key. Figures 4 and 5 illustrate an example of the topology for a 25 and 50 node simulation respectively.

4.2.1 One Subnetwork with 25 Nodes, Network Size of 25 Nodes

The first simulation was run with a network size of 25 nodes. The subnetwork size was also set to 25 nodes.

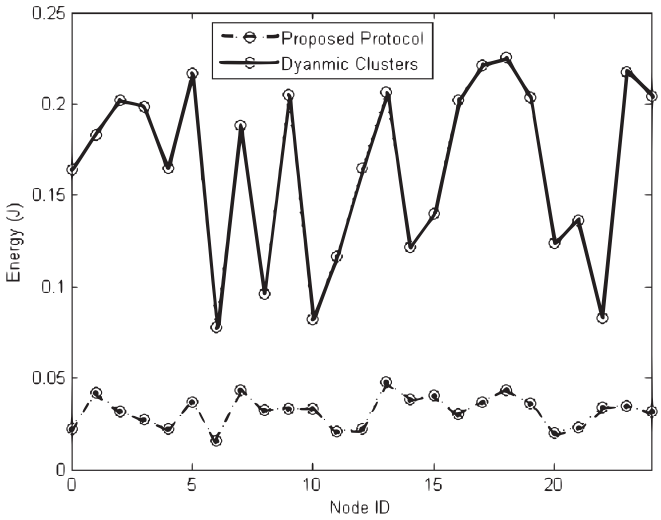

Figure 8: Energy used per node

Thus, in this simulation, it was assumed that the subnetwork included the entire network. The cluster heads for the network were nodes {15, 18, 24} for Clusters 1-3 respectively. Figure 7 shows the topology of this network.

After the nodes exchanged keys during the network initialization phase of the key management protocol, it was found that Cluster 1 (top left) had the highest average energy left. This is reasonable since the upper left area of the network topology is fairly sparse and not many HELLO messages were overheard in that area. Node 15 was designated the HCH and it was found that the cluster had enough nodes to fulfill all the needed partial keys, so no other clusters needed to be selected. Node 14 collected the 7 partial keys and broadcasted the subnetwork key to the entire subnetwork (network). For the DGKM protocol, the CH was chosen to be in the middle of the network (Node 19), and the entire network was used as a cluster. The energy per node was recorded and is shown in Figure 8.

It has been found the total energy used in the network for the proposed protocol was 1.05J, while for the dynamic key management protocol a total energy of 4.20J was used. Figure 8 depicts that for each node the EHKM utilizes less energy than the dynamic key management strategy. This is because only selected nodes in the subnetwork are responsible for generating a partial key and transmitting it to the HCH. This allows other nodes (not in the chosen clusters) to conserve energy. In this scenario, nodes {7, 8, 10, 13, 14, 15, 22} were in the chosen cluster and have higher energy consumption than the others. Additionally, EHKM receives necessary partial keys faster than the dynamic key management protocol since it required fewer transmissions to complete the task. In this simulation, it took 0.08s for Node 14 to receive the 7 partial keys. The dynamic key protocol took 0.63s to gather the required partial keys.

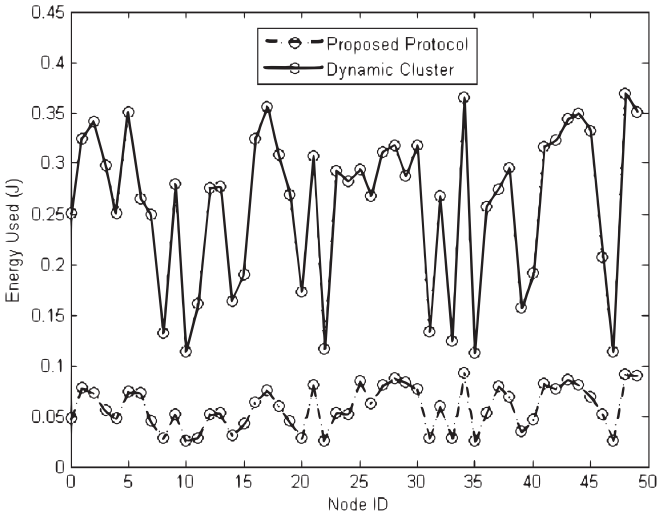

Figure 9: Energy consumption for 50 node network

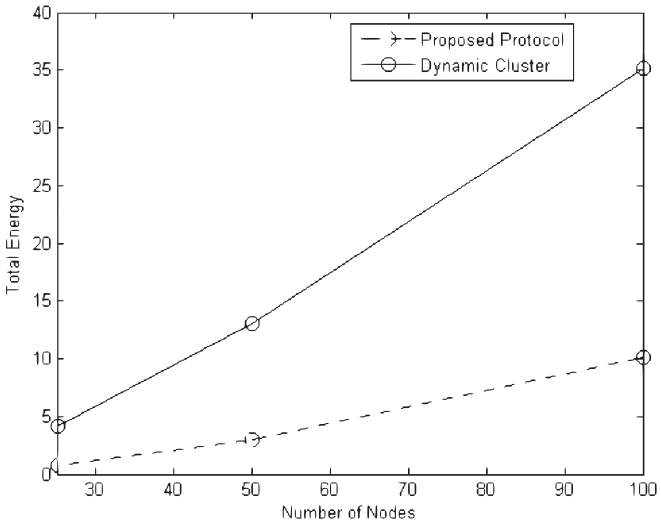

Figure 10: Energy utilized

4.2.2 One Subnetwork with 25 Nodes, Network size = 50 nodes

The next simulation was run with a 50 node network and a subnetwork of 25 nodes. The network was simulated from deployment until the successful creation of the first cluster keys. The event that is being monitored is assumed to be in the middle of the network and the SOS protocol is used to cluster the nodes and form cluster heads. For the DGKM protocol, two clusters of 25 nodes were utilized. The network energy consumption was then measured.

Figure 9 shows that EHKM requires less energy than the dynamic clustering algorithm for every node since only the nodes in the subnetwork are active for EHKM. Additionally, only necessary number of nodes in the subnetwok generates and transmits the partial keys. For the proposed EHKM protocol, the total energy consumption is equal to 2.06J, while the original dynamic cluster key management algorithm used 13.45J to create the cluster keys. Also in this scenario the EHKM completes the key generation quicker than the original protocol. The time needed to gather the partial keys was equal to 0.16s for the EHKM and 1.47s for the dynamic algorithm.

4.2.3 One Subnetwork with 25 Nodes Results Summary

The simulation was also run for a 100 nodes network with the analogous results. The summary of the results for all three network sizes is shown in Figures 10, 11 and 12.

The proposed version of the dynamic group key management protocol is more scalable than the existing dynamic group key management protocol in terms of energy consumed, delay and number of packets dropped. This is because the number of transmission needed to create the m desired partial keys is limited the minimum number of nodes. The DGKM protocol requires all the nodes in a cluster to generate a partial key even when the number

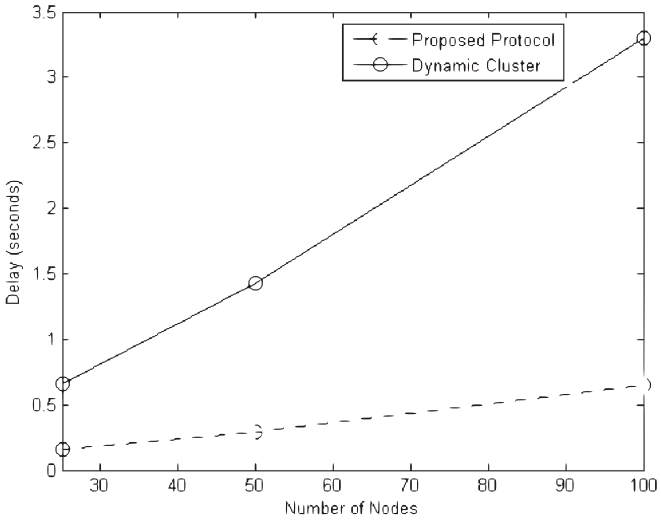

Figure 11: Delays

of nodes in the group may be much greater than m. This causes excess of partial keys to be sent to the cluster head and increases the energy usage with the generation and transmission of these partial keys.

The proposed protocol uses the subtree average energy function, thus the number of partial keys generated and sent to the subnetwork head is either equal to, or slightly greater than m. Not all nodes in the subnetwork generate and transmit partial keys. Additionally, also the nodes higher in the hierarchy are chosen to generate the partial keys to further minimize communication needed. In contrast, the DGKM uses only leaf nodes to generate the partial keys.

4.2.4 Multiple Events in One Network

In the previous sections, only one subnetwork was considered. In this section, a network is considered where multi-

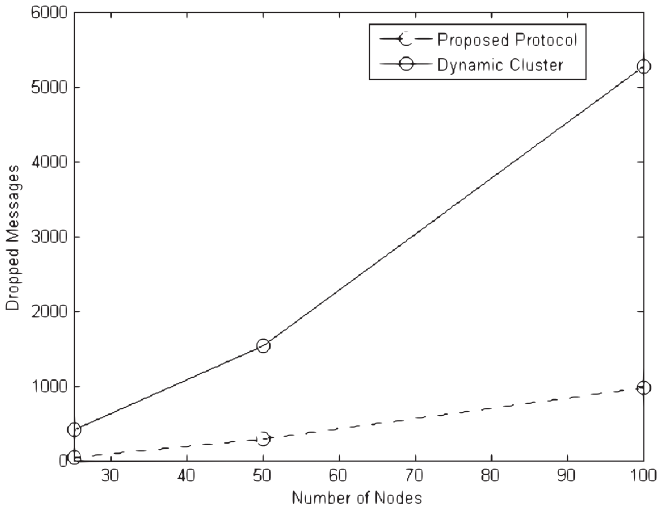

Figure 12: Dropped packets

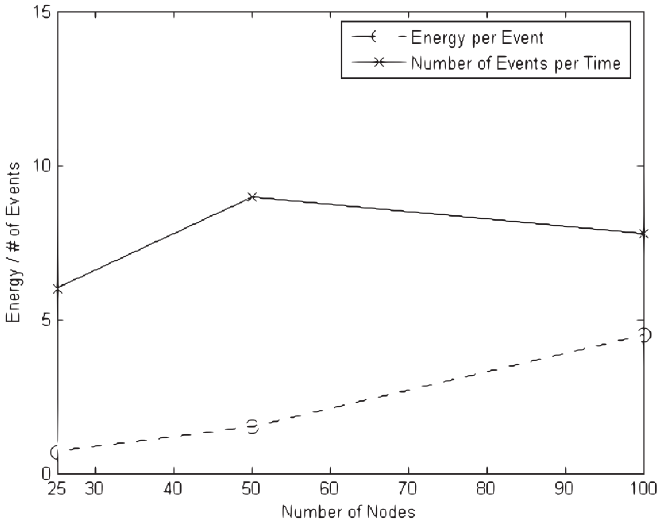

Figure 13: Energy and number of events per T

ple events take place over a period of time. In the dynamic group key management protocol, it is assumed that the groups would rekey regularly, using the same amount of energy that was consumed during the initial key establishment phase. The period of time between re-establishing keys is defined as T. In contrast, the EHKM protocol performs rekeying only when a new event is sensed. As a result, the EHKM will be more efficient than DGKM when the events occur seldom than period T. The simulation scenarios were varied in order to find out the maximum number of events that can occur per time period T for which the proposed protocol to still more energy efficient than the DGKM protocol. Several simulation networks of varying sizes were established where events periodically occurred within the network. The average energy for establishing a subnetwork key for these events was found. The results of these simulations can be found in Figure 13.

First it can be noticed that the average energy consumed to create the subnetwork key is less than the energy consumption presented in Figure 10. There are two reasons for this difference. First, the initial key establishment phase of individual keys is not needed for a pre-existing network. Additionally, many AODV routes are found in the first key establishment phase and need not be found again. Thus, more energy can be conserved since fewer route discovery tasks are required.

In Figure 13, the line with circle markers represents the average energy in Joules needed to create a key for the subnetwork dynamically generated in response to the event in the network. The line with x markers represents the number of events that could occur in time period T while EHKM continues to be more energy efficient than the DGKM protocol.

The number of events that can occur in time period T is above 6 for all tested network sizes. Additionally, it can be expected that events within a network will be both spatially and temporally related, for example cyclic

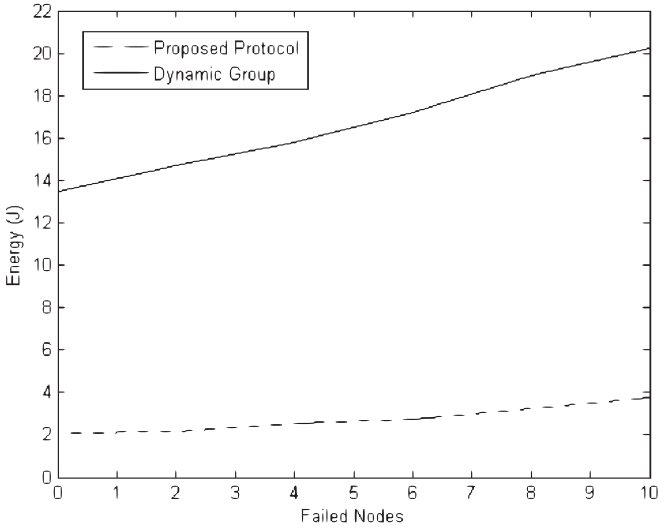

Figure 14: Energy cost of failed nodes

events. Thus, an event in a network will likely persist for sometime and then either propagate or disappear for a significant period of time. In this case, the ability to monitor more than 5 events in time period T with a consistent savings in energy is sufficient to conclude that EHKM will conserve energy over the DGKM protocol in terms of key creation and distribution over the lifetime of the network.

4.2.5 Node Failures

In this section EHKM is tested with random node failures in the network. The network size is set to 50 nodes and the number of nodes within the network that randomly fail is varied. The energy consumed and time to complete the protocols is measured.

Figures 14 and 15 illustrate the energy consumption and key setup time with varying number of failed nodes.

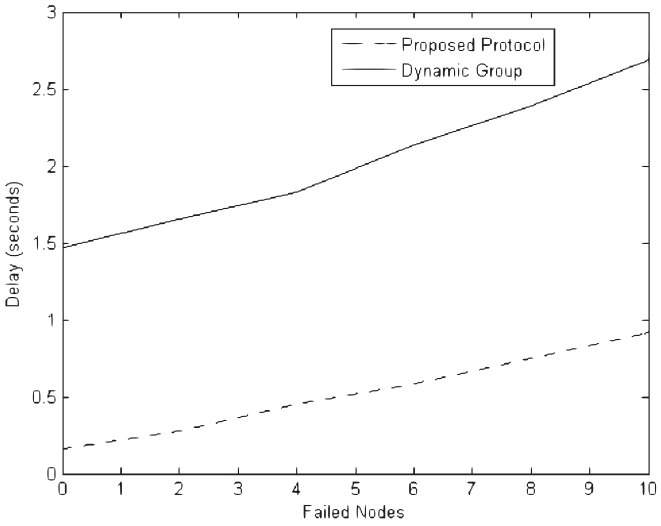

Figure 15: Delay with failed nodes

When compared with DGKM, the energy consumption for the proposed protocol increases slower with number of failed nodes since not every node is involved in the key management process. Therefore, some node failure will not affect the performance of the EHKM protocol at all and the network will continue to function normally. Additionally, fewer modifications to the routing table need to be made when a node failure does affect the routes, since the close proximity of active nodes reduces the damage to the nearest neighbors. Furthermore, due to the smaller numbers of nodes participating in EHKM key generation, there is a smaller chance that the failed node participates in many routes and case a major disruption.

5 Discussion of Security

The following section discusses some salient aspects of the security of EHKM in relation to the existing LEAP and DGKM algorithms. The security of a network can be measured in two ways. Assuming a secure encryption algorithm is paired with the key management algorithm, the main measurement of the security of a network will be the size of the network key. All other aspects of a network being the same, a network with a longer key size will remain secure for a longer duration of time interval than that with a smaller key size. The second measurement of the security of a key management scheme is the variety of attacks and their intensity to which the protocol is able to defend against and those that it is vulnerable to. This section attempts to discuss some of these attacks and the defenses the proposed protocol implements against them.

5.1 Network-Wide and Individual Keys

This section of the paper describes the security of the static aspect of the proposed protocol. It includes the network-wide key which is used for broadcast. And the nodes' individual keys used for localized communication.

When a sensor node is compromised it is assumed that the adversary gains knowledge of all the keying materials present in that sensor node. For EHKM this would include the broadcast key as well as the individual keys of the neighboring nodes and the unique key shared with the base station. The static aspect of this key management protocol limits the affected area of a compromised node to a very small portion of the network. A compromised node only gives the adversary the keys of nodes immediately neighboring the compromised node and the group key. In the case that this compromise goes undetected, only broadcast messages and local traffic will be compromised. For that reason, it is assumed that broadcast messages from the base station do not need to be confidential.

In the case that a compromise of a node is detected, an efficient method of revoking the node from the network and changing the group and individual keys of the neighboring nodes is needed.

5.1.1 Node Revocation

Changes in the group key may either be initiated by the base station either periodically or after notification that a node has been compromised. In either case, it is imperative that the group key change messages from the base station be authenticated to prevent forgery, replay and impersonation attacks. In this paper, we use the μTESLA broadcast authentication protocol proposed in [14]. In the μTESLA protocol, the controller creates a key chain and preloads the last value of the key chain in the nodes before deployment. The controller periodically releases the keys in its key chain in the reverse order they were created in. The period between key disclosures is called a μTESLA interval period. The base station can then broadcast messages encrypted or signed with the next key to be released. Nodes receiving a μTESLA packet buffer the packets until the next key is disclosed. This key can be authenticated through the next μTESLA key to be released and the message can be authenticated by the current key.

Fore this protocol let node n be a recently compromised node with neighbor set $N = \{m_1, m_2, ...m_i\}$ where m_1 through m_i are nodes that are neighbor to n. Node n is assumed to have the current group key as well as the individual keys of all the nodes in N. Once the base station has been notified of node n's compromise it sends out a broadcast message, X, to revoke node n from the network.

$$X : Controller \rightarrow * : n, f_{k'g}(0), MAC(k_{i+1}, n||f_{k'g}(0))$$

In message X, $k'g$ is the new group key and $f_{k'g}(0)$ is a pseudo-random function based on $k'g$ that all nodes possess. The value k_{i+1} is the next μTESLA key to be released. This message does not release the new group key, but provides the network with a method of revoking

a node via messages that can be efficiently authenticated. The value of $f_{k'g}(0)$ allows nodes to verify the key $k'g$ that they will receive in a future packet. The MAC attached to the message allows nodes to verify that this message came from the base station after the base station releases the key k_{i+1} to the network. Once the node has verified the message X as being from the base station, it stores the value $f_{k'g}(0)$ until it receives the new group key. All nodes that received a message X will immediately broadcast it to their neighboring nodes, flooding the network.

5.1.2 Secure Network Key Distribution

Once the base station is confident that all nodes have received the node revocation message it broadcasts a new group key message. The new group key, $k'g$ is encrypted using each node's individual key and broadcast to its neighbors. The neighbors verify the authenticity of the new group key using $f_{k'g}(0)$ then encrypt it using their individual key and broadcast it to their neighbors. This process continues through the network, except for nodes in the set N. Since node n has the individual keys for these nodes, these nodes do not pass the packet on immediately.

Nodes in the set of N first create a new random key to be their new individual key. They then wait a period of time, T_w, such that it is likely that a neighboring node that is also in set N has received the new group key. T_w could be set to be slightly greater than the average propagation delay between two arbitrary nodes in the network to allow the message to travel from a neighbor of the originating node to another node in N. Once this time has elapsed, the node broadcasts its new individual key encrypted using the new group key. Through this protocol the compromised node n is prevented from learning the new group key or the new individual keys of its neighboring nodes. Additionally, the base station can change the group key periodically by broadcasting the message X in the previous section without a node identifier. In this case, all nodes will get the new group key. Nodes are also free to change their individual keys at any time by encrypting their new individual key with the group key and sending the new individual key to its neighbors. If a node receives a node revocation message and has changed its individual key since the last group update, it must update its individual key using the new group key as soon as the new key has been authenticated.

5.2 Dynamic Key Management

The integrity of the dynamic key management aspect of EHKM depends on the security of the static aspect of the protocol since the partial keys and subnetwork keys are both encrypted using the individual keys of the nodes while the subnetwork key is being created. The subnetwork keys will be secure as long as the guidelines for maintaining the integrity of the individual keys are followed. Additionally, since the subnetwork key formation is localized, compromised nodes outside of radio communication range of the subnetwork will not be able to eavesdrop on the partial keys or the subnetwork key. In fact, these compromised nodes will not even be aware a subnetwork is being formed.

Since nodes only participate in the dynamic key management aspect of this protocol if they are actively involved in a subnetwork, a DOS attack is difficult to achieve in this network structure. Nodes can pretend to be sensing an event and attempt to create a subnetwork, but only other nodes that are sensing the attack will be become a part of the subnetwork. Nodes will also only reply to the *start_algorithm* messages during the creation of a subnetwork key so the subnetwork is safe from DOS attacks during that stage of the protocol.

6 Conclusions

In this paper a key management protocol was proposed which combined a static and dynamic key management approach to create a hybrid, energy-efficient, scalable key management scheme. The protocol utilizes the ability to group nodes into active subnetworks and passive groups and provide differing levels of security to the different groups. The active subnetworks utilize a dynamic key management approach while the passive or inactive nodes rely on static key management techniques. The use of dynamic key management protocols for only the active portions of the network decreases the amount of energy used and provides a more scalable approach to key management. Additionally a protocol for revoking a node from the network and for updating the network key and node's individual keys was proposed.

The scheme was analyzed for complexity and simulated in Ns2. It was compared to two schemes proposed in previous literature and shown to be more energy efficient during key generation and incurs lower delay and fewer packet losses. The protocol was shown to be more scalable than other protocols as network sizes get larger and nodes begin to fail.

References

[1] D. W. Carman, P. S. Kruus, and B. J. Matt, *Constraints and Approaches for Distributed Sensor Network Security*, NAI Labs Technical Report, no. 00-010, Sep. 2000.

[2] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *Proceedings of 2003 Symposium on Security and Privacy*, pp. 197-213, May 2003.

[3] D. Culler, D. Estrin, and M Srivastava, "Overview of sensor networks," *IEEE Computer Society*, vol. 37, no. 8, pp. 41-49, Aug. 2004.

[4] J. Deng, C. Hartung, R. Han, and S. Mishra, "A practical study of transitory master key establishment for wireless sensor networks," *Proceedings of*

First International Conference on Security and Privacy for Emerging Areas in Communications Networks, pp. 289-302, Athens, Greece, Sep. 2005.

[5] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A key predistribution scheme for sensor networks using deployment knowledge," *IEEE Transaction on Dependable and Secure Computing*, vol. 3, no. 1, pp. 62-77, Jan.-Mar. 2006.

[6] L. Eschenauer, V. D. Gligor, "A key-management scheme for distributed sensor networks," *Proceedings of the 9th ACM Conference on Computer and Communication Security*, pp. 41-47, Nov. 2002.

[7] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 60-96, Feb. 2004.

[8] D. Liu, and P. Ning, "Location-based pairwise key establishments for static sensor networks," *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 72-82, ACM Press, New York, NY, 2003.

[9] X. Li, Y. Wang, and O. Frieder, "Efficient hybrid key agreement protocol for wireless ad hoc networks," *Proceedings of 11th Intl. Conference on Computer Communications and Networks*, pp. 404-409, Oct. 2002.

[10] D. Liu, and P. Ning, "Improving key predistribution with deployment knowledge in static sensor networks," *TOSN*, vol. 1, no. 2, pp. 204-239, 2005.

[11] D. Liu, P. Ning, and R. Li, "Establishing pairwise keys in distributed sensor networks," *ACM Transaction Information System Security* vol. 8, no. 1, pp. 41-77, 2005.

[12] M. A. Moharrum, M. Eltoweissy, and R. Mukkamala, "Dynamic combinatorial key management scheme for sensor networks," *Wireless Communications and Mobile Computing 2006*, pp. 1017-1035, 2006.

[13] B. Panja, S. Madria, and B. Bhargava, "Energy and communication efficient group key management protocol for hierarchical sensor networks," *Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 1, pp. 384-393, Taichung, Taiwan, June 2006.

[14] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security protocols for sensor networks," *Proceedings of 7th Annual International Conference on Mobile Computing and Networks*, pp. 189-199, Rome, Italy, 2001.

[15] A. Price, K. Kosaka, S. Chatterjee, "A secure key management scheme for sensor networks," *Proceedings of the Tenth Americas Conference on Information Systems*, pp. 1739-1745, New York, Aug. 2004.

[16] S. Ratnaraj, S. Jagannathan, and V. Rao, "Self-organizing sensor sub-network protocol for structural health monitoring," *Proceedings of the SPIE*, vol. 6167, pp. 142-149, Feb. 2006.

[17] *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, NIST Special Publication 800-67 Version 1, 05, 2004.

[18] P. Tague, and R. Poovendran, "A general probabilistic model for improving key assignment in wireless networks," *WiOpt 2006*, pp. 251-259, 2006.

[19] P. Traynor, R. Kumar, H. Choi, G. Cao, S. Zhu, and T. La Porta, "Efficient hybrid security mechanisms for heterogeneous sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 663-677, June 2007.

[20] S. Zhu, S. Setia and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62-72,, Washington D.C., Oct. 2003.

Tim Landstra graduated from Fort Zumwalt North in O'Fallon, Missouri in May 2000. He attended the University of Missouri Rolla from August, 2000 until May, 2006. He received his Bachelor of Science degree in Computer Engineering from UMR in May, 2004 and his Master of Science degree in Computer Engineering from UMR in August 2006. His graduate studies in Computer Engineering with an emphasis on Secure Protocols in Wireless Sensor Networks were supported by a Graduate Assistance in Areas of National Need (GAANN) fellowship. He works in Sandia National Labs from 2006.

Sarangapani Jagannathan was born in Madurai, India and he received the Bachelor's degree in Electrical Engineering from College of Engineering, Guindy at Anna University, Madras India in 1987, the Master of Science Degree in Electrical Engineering from the University of Saskatchewan, Saskatoon, Canada in 1989, and the Ph.D. degree in Electrical Engineering from the University of Texas in 1994. During 1986 to 1987, he was a junior engineer at Engineers India Limited, New Delhi, as a Research Associate and Instructor from 1990 to 1991, at the University of Manitoba, Winnipeg Canada, and worked at Systems and Controls Research Division, in Caterpillar Inc., Peoria as a consultant during 1994 to 1998. During 1998 to 2001 he was at the University of Texas at San Antonio, and since September 2001, he is at the University of Missouri-Rolla where he is currently a Rutledge-Emerson Distinguished Professor and Director of NSF Industry University Cooperative Research Center on Intelligent Maintenance Systems. He has coauthored more than 220 refereed conference and juried journal articles and several book chapters and three text books entitled Neural Network Control of Robot Manipulators and Nonlinear Systems, published by Taylor & Francis, London in 1999, Discrete-Time Neural Network Control and Wireless Ad hoc and Sensor Networks; Performance and Control. His research interests include adaptive and neural network control, MEMS and embedded sensor networks, computer/communication/sensor networks, prognostics, and autonomous systems/robotics.

Dr. Jagannathan received several gold medals and scholarships. He was the recipient of Teaching Commendation award in 2005, Presidential Award for Research

Excellence at UTSA in 2001, NSF CAREER award in 2000, Faculty Research Award in 2000, Patent Award in 1996, and Sigma Xi Doctoral Research Award in 1994. He currently holds 20 patents and several are in process. He has served and currently serving on the program committees of several IEEE conferences. He is currently serving as the Associate Editor for the IEEE Transactions on Control Systems Technology and IEEE Transactions on Neural Networks. He is a member of Tau Beta Pi, Eta Kappa Nu, and Sigma Xi. He is a Senior Member of the IEEE.

Maciej Zawodniok was born in Knurow, Poland. Mr. Zawodniok graduated from the Silesian University of Technology with a Master of Science degree in Computer Science in 1999. He received a Ph.D. degree in Computer Engineering from the University of Missouri-Rolla in 2006. During 1999 to 2002, he worked for Ericsson AB as a consultant, where he delivered courses about cellular telecommunication for Ericsson Company and its customers around the world. Then, during 2002 to 2003 he worked as a Software Engineer in Motorola Polska. There, he was responsible for preparation and implementation of a novel automatic testing of cellular network's equipment. During 2003-2006, Mr. Zawodniok was a Research Assistant at University of Missouri-Rolla, and a Post Doctoral Fellow from 2006 to 2008. From 2008 he is at Missouri University of Science and Technology, where he is an Assistant Professor in Computer Engineering and Assistant Director of NSF I/UCRC on Intelligent Maintenance Systems.

Mr. Zawodniok's research focuses on adaptive and energy-efficient protocols for wireless networks, network-centric systems, network security, cyber-physical and embedded systems. Mr. Zawodniok is also involved in interdisciplinary projects related to application of RFID technology and wireless sensor networks in manufacturing and maintenance. He is a member of IEEE from 2004.