01 Oct 2007

# Energy-Efficient Hybrid Key Management Protocol for Wireless Sensor Networks

Timothy J. Landstra

Maciej Jan Zawodniok
*Missouri University of Science and Technology*, mjzx9c@mst.edu

Jagannathan Sarangapani
*Missouri University of Science and Technology*, sarangap@mst.edu



## Recommended Citation

T. J. Landstra et al., "Energy-Efficient Hybrid Key Management Protocol for Wireless Sensor Networks," *Proceedings of the 32nd IEEE Conference on Local Computer Networks, 2007*, Institute of Electrical and Electronics Engineers (IEEE), Oct 2007.
The definitive version is available at https://doi.org/10.1109/LCN.2007.135

# Energy-Efficient Hybrid Key Management Protocol for Wireless Sensor Networks

Tim Landstra, Maciej Zawodniok, S. Jagannathan
Department of Electrical and Computer Engineering
University of Missouri-Rolla, MO 65409, USA.
{tjland, mjzx9c, sarangap@umr.edu}

*Abstract*— **In this paper, we propose a subnetwork key management strategy 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. Results of simulations of the protocol in Ns2 are presented and the complexity of the protocol is analyzed. The proposed protocol reduces delay by 50% and energy consumption by 70% over the existing dynamic group key management (DGKM) scheme.**

***Keywords-component; wireless sensor network, key management, energy efficiency, network security***

## I. INTRODUCTION

Wireless sensor networks have recently come into the forefront of research due to their possible uses in military and disaster relief cases. Wireless sensor networks (WSN's) are a highly constrained type of network, comprised of sensor nodes with limited capabilities and larger gateway nodes with more capabilities. In sensor networks, the toughest constraints include the limited available energy and memory. Thus, lightweight and energy-efficient security protocols are necessary for these networks.

Some papers in literature investigate the use of dynamic key management techniques in WSNs [1]. Dynamic keys are preferable 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. In the paper by Panja et al. [1], a secure and dynamic group key management (DGKM) protocol is proposed utilizing a modified Tree Based Group Diffie-Hellmann key management protocol [2]. This protocol allows clusters in a sensor network to dynamically create a key for the cluster by having nodes 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 $\alpha^k \bmod p$ 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 $\alpha$ 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. 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 sensor networks to realize two distinct security and energy levels within the network. The protocol 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. 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. Additionally, we propose a method to combine a static and dynamic key management protocol to create a hybrid protocol.

The proposed protocol allows the network to form a high security subnetwork localized around a sensing event in the network. This aspect of the protocol relaxes the assumption of homogenous security needs of a network that is present in most literature on sensor network 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 affected cluster 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 the paper by Ratnaraj et al. [3] from a routing perspective and our protocol further improves security and energy efficiency.

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 [7-10] 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

0742-1303/07 $25.00 © 2007 IEEE
DOI 10.1109/LCN.2007.135

increases with network size [7-9]. A prior or explicit knowledge about deployment as used in [8] and [9] can only reduce the number of required pre-deployed keys, but it still will increase with network size. 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.

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 [7-8], or found among the pre-deployed keys [9-10], or established through a third party who shares the pair-wise keys with each of the nodes in the pair [7-10]. 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.

In this paper we analyze the proposed protocol and compare 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 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 and the Dynamic Key Management Protocol and provides a similar level of security.

## II. Sensor Network Architecture

Sensor networks are often deployed remotely and the network designers have no control over the location or placement of the sensors in the terrain, or the relation to other sensors around them. This paper considers 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 able to be compromised by an enemy and can safely exchange keys [4]. 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 the nodes 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 by Ratnaraj et al. in [3] 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 measure their proximity to the event by measuring the strength of the signal they are receiving. If the sensed signal strength is greater than a design threshold, then such nodes will group themselves into a subnetwork. Once the subnetwork has been established, the nodes exchange their energy remaining and a number of cluster heads are chosen based on energy remaining, proximity to the event, and size of the subnetwork. Nodes are then clustered by proximity to the chosen CH's.

Nodes are expected to know the number of nodes that are part of the same subtree or cluster and part of the same level as itself. The number of nodes that are part of cluster $i$ on level $j$. is denoted as $n_{ij}$. 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 [3] is used.

## III. Protocol Details

This protocol uses two separate key management schemes: for a group-wide and subnetwork key management. The first scheme manages a group-wide key and individual keys. 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.

### A. 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 [2]. 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.

**Remark 1**. The EHKM protocol considers any new node added after $T_{min}$ time as an intruder. This improves resilience to security attacks from approaching enemy at the expense of increased difficulty of adding a new legitimate node.

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:

$$Ni \rightarrow 1\text{-}hop : Node\_ID,\ E_{K2}(nonce),\ MAC(K2,\ Node\_ID \| nonce) \quad (1)$$

The node's HELLO message contains a random nonce encrypted with $K_2$, which serves as its secret key for any messages it sends during the network's lifetime. Any node receiving a HELLO message decrypts the nonce and verifies the message using the MAC. With a successful verification, the node stores the ID and nonce pair. Thus, whenever a node receives a non-broadcast message in the future from a neighbor, the node knows the packet was encrypted by the nonce that is paired with the source's ID and is able to decrypt the message. Once $T_{min}$ has elapsed, the nodes are responsible for deleting $K_2$ from their memories.

**Remark 2**. Key $K_2$ is used during the $T_{min}$ time to authenticate the nodes. The $T_{min}$ is set to be lower than a minimum time required for attacker to compromise a node. Hence, the attacking node will not know the $K_2$ key and thus fail the authentication.

### *B. Subnetwork Key Management Protocol*

The subnetwork section of the key management protocol begins with the creation of a subnetwork within the WSN. First, the cluster heads find the average energy remaining for all the nodes in their cluster.

$$avg\_energy(i) = sum(E_{ij}) / n_i \quad (2)$$

where *avg_energy(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 $i^{th}$ 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 Head Cluster Head (HCH) is selected among the cluster head of the subnetwork. The cluster head with the highest average energy left in its cluster becomes HCH.

$$HCH = CH(\max(avg\_energy(i))) \quad (3)$$

The network has a predetermined number of partial keys that are sufficient to create a subnetwork key that is determined a priori. This value, *m*, has been suggested to be 15 in past literature [1], but 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. Once the HCH has been determined, the other cluster heads check to see if the number of nodes in the HCH's cluster, $n_{HCH}$, is sufficient to generate *m* partial keys. If it's not, then the CH with the second highest average energy remaining in its member nodes is added as chosen cluster. This method continues until the number of nodes in the chosen clusters is greater than *m*. Once sufficient cluster heads have been chosen to generate the *m* partial keys that are needed, the algorithm is started. The cluster heads broadcast a *start_algorithm* message to their cluster 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 a cluster's 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 ($m - sum(n_{other\_clusters})$). That is, the number of partial keys desired, minus the number of nodes in all the other selected clusters. This ensures that only exactly *m* partial keys are returned to the HCH thus maximizing energy conservation.

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 if the node is not a leaf node. If the node is a leaf node, then it creates its partial key and sends the key to its parent and its CH. If the depth field does not equal -1, the node subtracts the number of nodes in its level and cluster and if the depth field is still greater than 0, rebroadcasts the message. If, after the subtraction, the depth field becomes less than or equal to 0, then the node acts as a leaf node and randomly creates its partial key and sends the key to its parent and to the HCH.

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 of the nodes as it is transmitted throughout the subnetwork.

### *C. Dynamic Cluster Key Management Algorithm*

Once the network determines the clusters that will be creating the partial keys and passing them to the head cluster head, a variant of the dynamic group key management protocol proposed in [1] by Panja et al. 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

$$f(k1, k2) = \alpha^{k1 \oplus k2} \bmod p \quad (4)$$

where *p* is a prime number, $\alpha$ is a primitive root of *p*, and $k_1$ and $k_2$ are the children's partial keys ($k_1, k_2 < p$). 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. Then, it broadcasts the key to the cluster.

The proposed protocol, unless otherwise specified, uses this algorithm for subnetwork key creation and distribution. Several modifications are made to the protocol to allow for more node topologies. One modification is that the 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.

**Remark 3**. In general, the traffic analysis attacks are a considerable threat to hierarchical schemes. The proposed EHKM counteracts such an attack by limiting the number of exchanged messages which can hint the location of CHs.

Additionally, the rotation of CHs in the clustering scheme ensures that traffic patterns change often. Moreover, the CHs rotation will facilitate quick recovery in case of physical destruction of a CH identified via the traffic pattern attack.

## IV. Performance Analysis

In this section, a performance analysis of the proposed protocol is performed. 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 number of node failures are analyzed.

### A. Complexity Analysis

Table I displays the results of the complexity analysis on the protocol and compares it to several existing key management schemes. As can be seen the message cost and complexity of the algorithm are both less than other schemes such as LEAP and dynamic key management. Note, that for most cases m << N.

TABLE I. Complexity Analysis

| | SetUp | Key Management | | Storage |
|---|---|---|---|---|
| | *Messages* | *Messages* | *Complexity* | |
| Leap [4] | N(1+5d) | 2(d-1)^2/N+2N | O(d^2) | (3d+2+L)KL |
| Dynamic [1] | - | 3N | O(logN) | mlogN+k |
| Proposed | N | $3m+N_S+N_{CH}$ | O(logm) | (d+2)KL+k |

where:

- d – average degree of connectivity of the network
- N – number of nodes in network
- NS – number of nodes in a subnetwork NS < N
- M – number of desired partial keys
- KL – individual key length (in bytes)
- K – dynamic subnetwork key length (in bytes)
- L – length of key chain
- NCH – Number of cluster heads in subnetwork

In Table I, the degree of the network is 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 the proposed protocol, 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 the proposed protocol 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 $N_{CH}$ cluster heads must distribute the average energy left in their clusters. This results in $N_{CH}$ messages. Then the *start_algorithm* message is distributed to *m* nodes who reply, both to their parent and to their cluster head. This causes 3*m* messages to be sent. Finally, the subnetwork key is broadcasted to the $N_C$ nodes of the subnetwork. The time complexity of this stage is log(*m*) since the *3m* messages dominates the $N_C$ and $N_{CH}$ 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 storage of the proposed protocol is measured in terms of how many bytes of storage must be used to store the keys of the key management protocol. For the proposed protocol, a node must store 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 of these keys have length *KL*. The proposed protocol 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.

### B. Simulation Results

The proposed protocol was simulated in Ns2 and compared to the dynamic key management protocol. The simulation results are shown in the following subsections. For the simulations, 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 $2.5 \times 10^{-5}$ nodes / $m^2$. 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. The routing protocol used was AODV. The networks were simulated from network deployment until the establishment of the subnetwork's first key. The simulations were repeated with random node placement and the results averaged. The performance of the proposed EHKM protocol has been compared with another group key-based scheme the DGKM [1].

*1) One Subnetwork with 25 Nodes, Network Size of 50 Nodes*

This 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 subnetwork key. 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. Three clusters were created from the 25 node subnetwork in range of the sensed event. For the SDGK protocol, two clusters of 25 nodes were utilized. The network energy consumption was then measured.

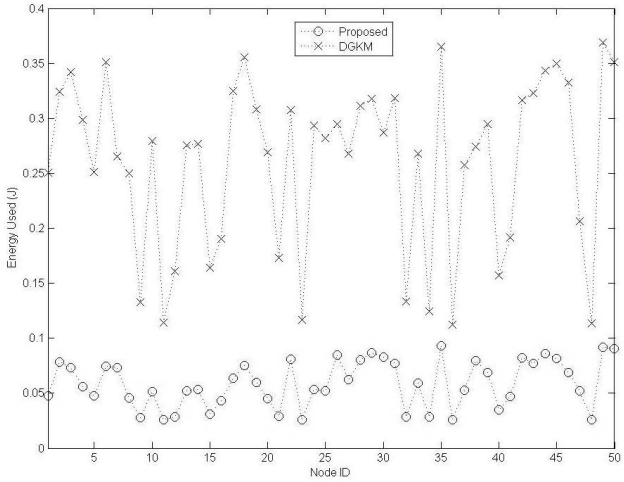


Figure 1. Energy consumed per node

For the simulation it was found the total energy used in the network for the proposed protocol was 2.73J and for the dynamic key management protocol a total energy of 14.2J was used. It is obvious from Figure 1, that the proposed protocol uses significantly less energy than the dynamic key management strategy. This is because not every node in the subnetwork is responsible for generating a partial key and transmitting it to the HCH. This allows nodes not in the chosen clusters to conserve energy. Additionally, the proposed protocol receives the necessary partial keys faster than in the dynamic key management protocol. 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.

*2) One Network with 25 Nodes Results*

The above simulation was run for a 25 and 100 node network as well. The summary of the results can be found below in Figures 2, 3 and 4. For the DKGM protocol, the network was broken up into *N/25* clusters of 25 nodes each.

As can be seen in Figures 2 through 4, the proposed subnetwork modification of the dynamic group key management protocol is more scalable to larger network sizes 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 packets to create the m desired partial keys is minimized by only involving the minimum number of nodes. The DGKM protocol requires all the nodes in a cluster to generate a partial key even when the number of nodes in the cluster may be much greater than *m*. This causes excess partial keys to be sent to the cluster head and increases the energy usage with the generation and transmission of these partial keys.

With the proposed protocol combined with the subtree energy average function, 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, nodes higher on the hierarchy are chosen to generate the partial keys to further minimize communication needed.

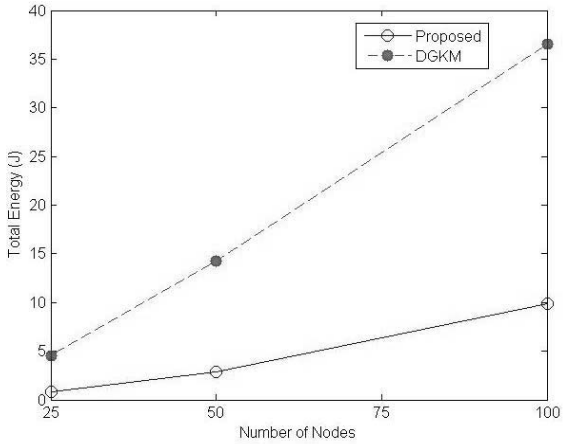


Figure 2. Energy utilized

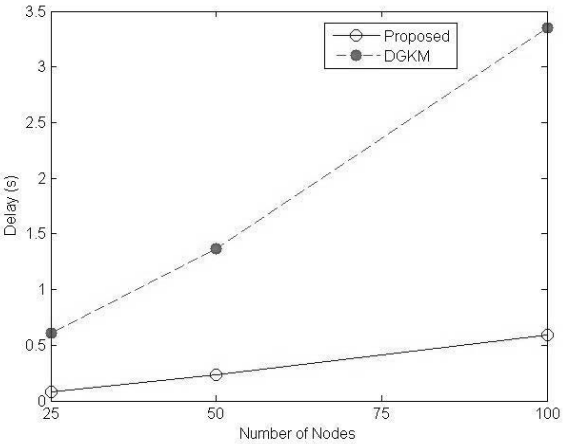


Figure 3. Delay

## C. Multiple Events in One Network

In the previous sections, only one generation of subnetwork was considered. In this section, a network is considered where multiple 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*. Since the proposed protocol differs from the DGK protocol by rekeying every time a new event is sensed, it is of interest to find how many events can occur per time period *T* to allow the proposed protocol to continue to be 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 5.

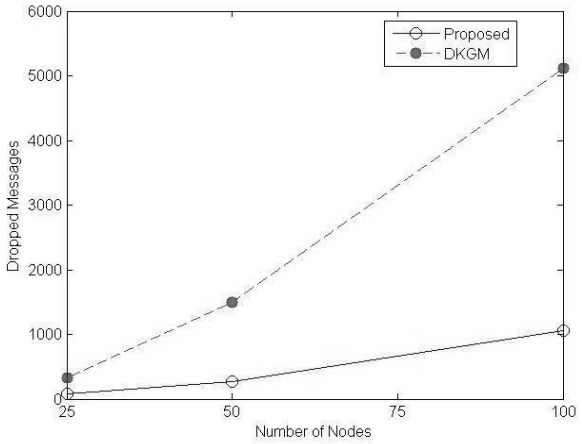


Figure 4. Dropped packets

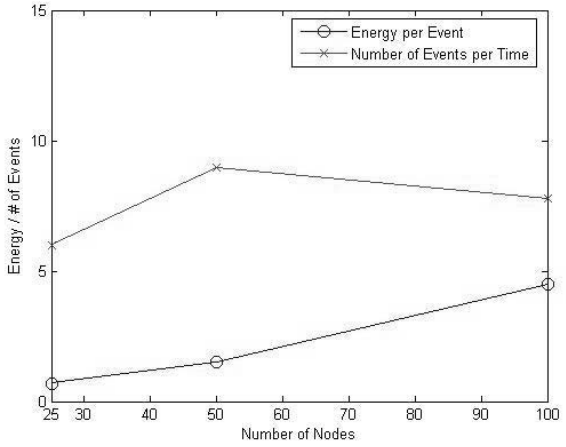


Figure 5. Energy and number of events per T

It can be seen that the average energy consumed to create the subnetwork key is less than the energy consumed in Figure 2. The reason for this is two-fold. 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 route discovery tasks are not required.

In Figure 5, the line with circle markers represents the average energy in Joules needed to create a key for the subnetwork dynamically generated to monitor the event in the network. The line with x markers represents the number of events that could occur in time period T and have the proposed protocol continue to be more energy efficient than the DGKM protocol.

As can be seen, the number of events that can occur in time period T is above 5 for all network sizes tested. It is proposed that events within a locale of a network will be both spatially and temporally related. 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 the proposed protocol will conserve energy over the DGKM protocol in terms of key creation and distribution over the lifetime of the network.

### *D. Node Failures*

In this section the proposed protocol 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 the time-to-complete the protocols are both measured.

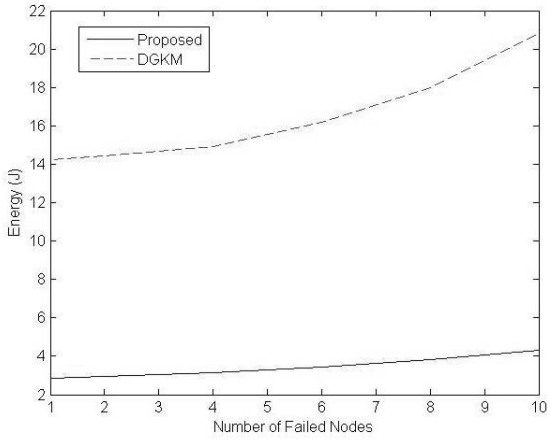


Figure 6. Energy cost of failed nodes

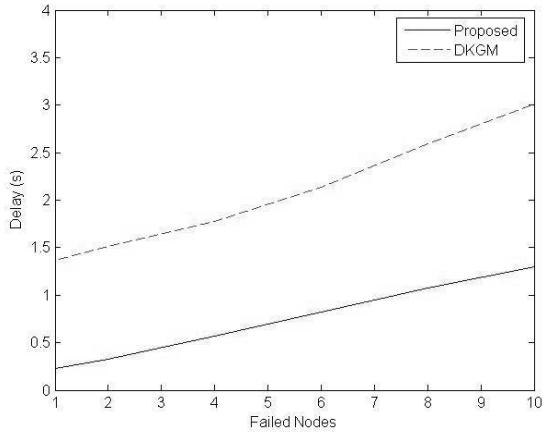


Figure 7. Delay with failed nodes

As can be seen from Figures 6 and 7, while both protocols are similarly resilient to failed nodes, the proposed protocol is slightly more robust in the presence of node failures. This is due to the fact that not every node is involved in the key management process. Therefore, there is a chance that a node failure will not affect the performance of the protocol at all and the network will continue to function normally. Since the nodes that participate in the key management protocol are physically close to each other, when a node failure does affect the routing of a message, fewer modifications to the routing table need to be made. Due to the smaller numbers of participating nodes, there is a lesser chance that the failed node has many layers of routing table below itself.

## V. SECURITY

The following section discusses some salient aspects of the security of the proposed protocol. 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 equivalent, 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 attacks 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.

### *A. Network-wide and Individual Keys*

This section of the paper describes the security of the static aspect of the proposed protocol. This aspect of the protocol includes a key common to all nodes in the network used for broadcasting messages as well as the node's 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 the proposed protocol 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 network key and individual keys of the neighboring nodes is needed. For our protocol a modification of the node revocation described in the LEAP protocol [4] is proposed.

#### *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 by Perrig *et al*. [6]. 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.

For this protocol, let *n* be the compromised node with set of neighbors $N=\{m_1, m_2, \ldots m_i\}$. 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.

$$X : Controller \rightarrow * : n, f_{k'g}(0), MAC(k_{i+1}, n \| f_{k'g}(0)) \quad (5)$$

where $k'_g$ is the new group key that will be released in a future message, $f_{k'g}(0)$ is a pseudo random function based on $k'_g$, which all nodes possess and the value $k_{i+1}$ is the next μTESLA key to be released. The MAC allows nodes to authenticate the message as originating from the base station when the subsequent μTESLA key is released. Once a node has authenticated the message, the value $f_{k'g}(0)$ is stored until the new group key is received.

#### *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 broadcasted to its neighbors. The neighbors verify the authenticity of the new group key using the previously stored $f_{k'g}(0)$ then encrypt it using their individual key and broadcast it to their neighbors. This process continues 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 random key to be their new individual key. They then wait a period of time, $T_w$, such that it is likely that the neighboring nodes that are also in set N have 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.

### B. *Dynamic Key Management*

The integrity of the dynamic key management aspect of the proposed protocol 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.

Nodes only participate in the dynamic key management aspect of this protocol. Consequently, 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.

## VI. 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] B. Panja, S. Madria, and B. Bhargava . “Energy and Communication Efficient Group Key Management Protocol for Hierarchical Sensor Networks,” Proc. of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2006), (Taichung, Thailand), pp. 384-393, June 2006.

[2] Y. Kim, A. Perrig, G. Tsudik. “Tree-Based Group Key Agreement.” ACM Transactions on Information and System Security (TISSEC), vol. 7, no. 1, pp. 60-96, Feb 2004.

[3] S. Ratnaraj, S. Jagannathan, and V. Rao, “Self-Organization Using Sub-Network Formation for Wireless Sensor Networks,” MS Thesis for S. Ratnaraj, 2005, University of Missouri ~ Rolla.

[4] S. Zhu, S. Setia and S. Jajodia, “LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks,” Proc. of the 10th ACM Conference on Computer and Communications Security (CCS ’03), pp. 62–72, Washington D.C., October, 2003.

[5] L. Eschenauer, V. D. Gligor. “A Key-Management Scheme for Distributed Sensor Networks,” Proc. of the 9th ACM Conference on Computer and Communication Security, pp. 41-47, Nov. 2002.

[6] A. Perrig, R. Szewczyk, V. Wen, D. Culler, D. Tygar, "SPINS: Security Protocols for Sensor Networks," Proc. of 7th Annual Intl.Conference on Mobile Computing and Networks, Rome, Italy, pp. 189-199, 2001.

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

[8] 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, Issue 1, pp. 62-77, Jan.-March 2006.

[9] 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 (Fairfax, Virginia), pp. 72-82, SASN '03, ACM Press, New York, NY.

[10] Patrick Traynor, Raju Kumar, Heesook Choi, Guohong Cao, Sencun Zhu, Thomas La Porta, "Efficient Hybrid Security Mechanisms for Heterogeneous Sensor Networks," IEEE Transactions on Mobile Computing ,vol. 6, no. 6, pp. 663-677, June, 2007.