An Energy-efficient Approach to Securing Distributed Sensor Networks (DSN)  
(Extended Summary)

Fei Hu*    Jason Tillett**    Sunil Kumar***

* IEEE Member {fei.hu@ieee.org}, Computer Engineering Department, Rochester Institute of Technology, Rochester, New York 14623, USA
** Senior Researcher {jtillett@netsup.net}, Laboratory for Autonomous Cooperative Microsystems College of Engineering, RIT
*** IEEE Senior Member {skumar@clarkson.edu}, Computer Engineering Department, Clarkson University, Potsdam, NY 13699

ABSTRACT --- Recent advances in micro-electro-mechanical systems (MEMS) technology, wireless communications, and digital electronics have enabled the development of low-cost, low-power, multifunctional sensor nodes. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes. On the other hand, confidentiality, integrity, and authentication services are critical to preventing an adversary from compromising the security of a distributed sensor network. Key management is likewise critical to establishing the keys necessary to provide this protection. However, providing key management is difficult due to the ad hoc nature, intermittent connectivity, and resource limitations of the sensor network environment. In this paper, the main performance requirements for securing ad hoc sensor networks are analyzed. Based on those requirements, we suggest a self-organized sensor network architecture that can provide flexible key management scheme. We further discuss an energy-efficient approach to re-keying in such a scalable network topology. Our conclusion is that the security problem in sensor networks cannot simply follow existing security schemes in typical networks such as general ad hoc networks and Internet since sensor networks has its own features such as very limited energy, large amount of nodes and frequent routing change.

Index Terms --- Sensor networks, wireless networks, security

1. Introduction

Progress in wireless networking and micro-electro-mechanical systems (MEMS) are contributing to the formation of a new computing domain – ad hoc sensor networks [20]. Unfortunately there is little research work carried out in securing ad hoc sensor networks ([2] being the exception). There are some solutions for securing general ad hoc networks in the literature [3-8]. However, those solutions cannot be applied efficiently in sensor networks where there can be thousands of energy-limited notes with very small local memory. For example, in [9] a solution was proposed to protect routing from attacks such as DoS (Denial of Service) by forged routing information maliciously injected in the network. Such a scheme is too heavy for the sensor nodes, both from the computational point of view and the storage point of view. The pioneer work on securing sensor networks proposed in [12] introduces “SPINS: Security Protocols for Sensor Networks” comprised of Sensor Network Encryption Protocol (SNEP) and µTESLA. The function of SNEP is to provide confidentiality (privacy), two-party data authentication, integrity and freshness. Our proposal to be described in this paper differs from SPINS in two fundamental and essential ways:

1) We focus on addressing the fundamental problem in securing ad hoc sensor networks: re-keying for each transmission of data traffic based on an energy-efficient network architecture;
(2) We consider the key management under the *dynamic ad hoc* sensor network architecture. In other words, a *self-organized* network topology-forming scheme is adopted for applying re-keying strategy.

### 2. Self-organized network architecture for re-keying

Our work is an extension of [18] and considers more the performance requirements of DSN. Like [18], we also assume all network nodes share a secret group identity key $k_{GI}$ that is used to provide authentication and to derive additional keys used to *provide confidentiality*. The group identity key’s crypto-period lasts for the whole duration of the network while the keys used to provide confidentiality are updated on regular intervals.

The DSN’s lifetime is divided up in ‘rounds’. Each round consists of three phases as illustrated in Figure 1. During the ‘normal’ phase the nodes use the Group Identity key $K_{GI}$ to provide authentication and message integrity, and a Traffic Encryption key $K_{TE}$ to provide confidentiality. To enable the distributed update of the traffic encryption key, the network is self-organized into a hierarchy architecture in ‘self-organization’ phase that will be discussed later.

Each hierarchy layer consists of multiple super nodes. During the ‘re-keying’ phase one of the super node is elected as key manager and is responsible for generating a new traffic encryption key and distributing it to the other super nodes. Each of the super nodes then distributes the new traffic encryption key to its cluster members.

![Figure 1. Networking working Phases](image)

Different from the approach of [18], we adopt a multiple-layed hierarchy networking architecture instead of a simple two-tie cluster architecture. Although there are many wireless network architectures that are based on flat many-to-many topology [21,22], we think that hierarchy organization is a better technique due to its following advantages:

- The size of the routing table in each of the nodes is reduced [23], [24], [25].
- The communication overhead for distributed routing is reduced [24].
- Another advantage of organizing the nodes into multiple layers is that less overhead is incurred for tracking mobile users [23].

Here we focus on a multiple-layer hierarchy networking architecture. Given large amounts of sensor nodes, we expect that they can form a multiple-layed self-organized architecture where the lowest layer consists of regular nodes and the higher-layers (>1) consist of ‘super nodes’. Occasionally new nodes can be added to sensor networks to compensate for ‘dead’ nodes that run out of energy. In other words, self-organization will happen periodically. The procedure of the self-organization can be formulated as follows:

Given a disk graph $G = (V, E)$, find a series of clusters for each connected component of $G$: $\{V_i \in V \mid i = 1, 2, 3, \ldots, M\}$, so that:

1. All nodes belong to a cluster in one of the layers. That is: $U_{i=1}^M V_i = V$.
2. Any subgraph of $G$ induced by each cluster is connected.
(3) Any of two clusters have a small constant number of common nodes. That is:
\[ |V_i \cap V_j| \sim O(1). \]

(4) Any node belongs to a constant number of clusters. That is:
\[ \text{If } S(v) = \{V_i \mid v \in V_i\}, \text{ then } |S(v)| \sim O(1) \]

(5) The size of each cluster is bounded. (i.e., the number of nodes in each cluster is bounded.) Assume K is an integer and less than the size of the DSN, we let: \[ K < |V_i| < 2K \].

The above problem is actually to find a rooted spanning tree of the graph [27]. One could use a Breadth-First-Search (BFS) tree, or any other tree. The main advantage of a BFS tree is that it has a radius, which is bounded by diameter of the graph.

3. Re-keying scheme based on the scalable architecture

We can use a simple distributed version of the above algorithm for re-keying protocol in sensor networks. The re-keying protocol is comprised of the following phases:

**Phase A triggering condition:**
- In the beginning of the deployment.
- Nodes are dead because of no-energy;
- New nodes are added for increasing sensing density;
- In some applications nodes have a certain mobility so that the network topology is changed.

**Phase A:** Self-organization phase: discuss later.

**Phase B:** Normal operation phase. The data traffic is encrypted with a generated key.

**Phase C triggering condition:**
- After a certain period \( T \), a new key should be generated for encrypting data traffic;
  - The value of \( T \) depends on the desired security level. The smaller the value of \( T \), the higher security level the sensor networks.

**Phase C:** Key-updating phase: discuss later.

In each phase, key generation can be initiated by any node in the network (the initiator will be the root of the BFS tree). If multiple nodes initiate Cluster Creation at the same time, simple tie breaking heuristics (e.g. initiator with least ID) is imposed to allow only one instance to proceed; the rest are not propagated.

In **Phase A**, the super nodes could be predetermined because of its higher power and larger memory. In some cases, there is no difference from the physical structure point of view between super nodes and regular nodes. In order to apply our self-organization algorithm, we need to adopt the weight value of each node to select super nodes. Such weights can account for node features such as mobility, battery level, distance from the other nodes and many others. As introduced in [18,29], the following expression can be used to calculate the nodes’ weight:
where the \( C_i \) is the constant weighting factors for the interest parameter \( P_i \).

In Phase A, first the system needs to perform “Self-organized neighbor discovery”. Each node, \( u \), transmits a tree discovery message, \( x \), which indicates its shortest hop-distance to the root, to its neighbors. If any neighbor of \( u \) receives this message and discovers a shorter path to the root through \( u \), it will update its hop-distance to the root appropriately and will choose \( u \) to be its parent. Message \( x \) should be encrypted using the key as follows:

\[
K_i = \text{hash}(K_{i-1})
\]

That key is actually obtained by applying a one-way function to the backbone key \( K_{i-1} \) generated in the previous re-keying period (the (i-1)th) and initialized as \( K_0 \). When message \( x \) is encrypted in node \( u \), it will be transmitted to its neighboring nodes with the following encryption format:

\[
E_{K_i}(w_u | x | MAC(K_0, < w_u | x >))
\]

Where \( MAC(K_i, < \cdot >) \) is a keyed Message Authentication Code (MAC) applied to message \(< > \) using the key \( K \). The symbol | indicates the concatenation of two messages. \( N_u \) means the neighboring nodes of node \( u \). We can thus guarantees the integrity of the message and the authenticity of its origin through the using of the group key to compute the MAC [18].

Once each node has the information on their neighbors, the super nodes can then use a new Cluster Key \( K_C \) to communicate with its nodes via the following message:

\[
\text{superNode} \Rightarrow \text{members} : E_{K_C}(x | K_C | MAC(K_0, < x | K_i >))
\]

In Phase B, the system is guaranteed to transmit traffic with confidentiality and authenticity. Here we need to distinguish two types of traffic: Data traffic and Control traffic. By data traffic we mean all the traffic on the network, be it routing or application-oriented traffic, as opposed to control traffic, by which we mean messages exchanged during the key management protocol. Data traffic is encrypted by all nodes using the same key, the Traffic Encryption Key, which changes during the network lifetime in order to protect the system from cryptanalytic attacks and to cope with changes in the cluster due to node extinction.

In Phase C, we need to handle the maintenance of our tree when each time new node joins or existing node leaves. Here we adopt the approach in [27] and relax the upper bound of the cluster size to \( 3k \) instead of \( 2k \) in order to ensure that we maintain connectivity in each cluster. A new node \( q \), on joining the network, establishes its set of neighbors, \( N(q) \). If any node \( d \in N(q) \) belongs to some cluster of size \( < 3k-1 \) then we add \( q \) to the cluster that \( d \) belongs to.

Periodically when re-keying executes, the most a super node can check is whether it can be a Potential Key Manager (PKM) based on its weight being larger than its neighbor super nodes. Then a PKM node, \( g \), generates the new Traffic Encryption Key, \( TEK_i \) (ith re-keying round), and broadcasts it to the other super nodes. Any message during this process will be encrypted using the current key \( K_i \):

\[
g \Rightarrow \text{Neighbor} - \text{superNode} : E_{K_i}(x | w_g | TEK_i | MAC(K_0, < x | w_g | TEK_i >))
\]

Since the keys used for its distribution and for the cluster forming are derived from \( K_0 \) by applying hash function for an adequate number of times, there is no need to distribute them. Each node can recompute the key it needs based on the counter of the number of times it has executed the protocol.

4. Initial simulation results
To verify our self-organization algorithm, we use simulation to investigate the clustering behavior of a 100-node sensor network. First we assume those nodes to be non-uniformly distributed in a 100m x 100m square area. We bound each cluster size from 10 ~ 20 nodes. Figure 2 shows the clustering results.

In Figure 3, the circled nodes are the finally selected super nodes based on its weight values. We found the 87% of the time, a cluster-head was correctly distributed in each natural cluster. Only 13% of the runs resulted in 2 cluster-heads being assigned to the same natural cluster. On less than 1% of the runs, more than 2 cluster-heads were assigned to a single natural cluster.

In Figure 4, we use a uniformly distributed topology instead of non-uniform distribution as in Figure 3. The size of each cluster is 10 ~ 20 nodes. We can see that our algorithm can still converge to multiple clusters quickly. In Figure 6, we change the size of the cluster and investigate the converge speed of our algorithm. If the size of each cluster is bounded below 5 nodes, our algorithm needs only 50 ~ 60 steps to converge to stable cluster architecture. The bigger cluster size, the faster our algorithm.

Figure 4 Convergence steps for clustering different set of sensors

References (omitted due to space limitation)