SPECTRA: Secure Power-Efficient Clustered-Topology Routing Algorithm in Large-scale Wireless micro-Sensor Networks

Fei Hu [1], Waqaas Siddiqui [1], and Xiaojun Cao [2]

[1] Department of Computer Engineering
Rochester Institute of Technology
Rochester, NY USA
E-mails: fei.hu@rit.edu, was6004ta@yahoo.com

[2] Department of Information Technology
Rochester Institute of Technology
Rochester, NY USA
E-mail:cao@it.rit.edu

Abstract

Wireless Sensor Networks (WSNs) have emerged as one of the hottest fields today due to their low-cost, self-organizing behavior, sensing ability in harsh environments, and their large application scope. One of the most challenging topics in WSNs is security. It is critical to provide confidentiality and authentication in order to prevent information from being compromised. However, providing key management for confidentiality and authentication is difficult due to the ad hoc nature, intermittent connectivity, and resource limitations of the sensor network. Though traditional public key-based security protocols do exist, they need large memory, bandwidths and complex algorithms, and are thus unsuitable for WSNs. To reduce the consumption of resources (energy, memory, CPU calculation time, etc.), it is necessary to use symmetric-key-based security. Current solutions to the security issues in WSNs do not consider the correlation between "routing" and "security" effectively. The focus of this work is on the integration of routing and key management to provide an energy efficient security and routing solution. Towards this goal, this work proposes a security protocol that encompasses the following features: integration of security and routing, dynamic security, robust re-keying, low-complexity, and multiple levels of encryption. Compared to many others such as SPINS [9], our security scheme with enhanced reliability and scalability consumes much less energy.

Keywords: Wireless Sensor Networks (WSN), Network Security, Ad hoc Networks, Cluster-based routing protocol

I. Introduction

More recently, a paradigm shift occurred from traditional macro-sensors to the micro-sensors used in Wireless micro-Sensor Networks (WSNs). A WSN is comprised of wireless sensor modules, called nodes. Each node is made up of a few key components: a micro-sensor to detect the desired event; a low-cost application-specific microprocessor; memory to store information; a battery; and a transceiver for communication between the node and the rest of the network.

Due to the nature of wireless communication, data is available in the air for any third party to acquire. This feature along with the ad hoc nature, intermittent connectivity, and resource limitations of WSNs result in a number of design challenges. For example, the availability of data to third parties could cause numerous disasters in many military or homeland security applications. Therefore, it is critical to provide confidentiality and authentication while preventing data information from being compromised. Traditionally, security is provided through public-key based protocols. However, these protocols require large memory bandwidth and complex algorithms [9]. The limited resources of WSNs make this type of security schemes unsuitable for implementation. Thus, security protocols that provide security while taking into account the unique features and resource limitations of WSNs are preferred

Currently, very limited work has been done on WSN security [9-15, 20-23, 29-30]. The pioneering work on securing WSN end-to-end transmission is SPINS [9] which requires time synchronization among sensors. It also proposed μTESLA, an important innovation for achieving broadcast authentication of any messages sent from the base-station (BS). An improved multi-level μTESLA key-chain mechanism was proposed in [29, 30]. Authors in [10] suggested a key-pool scheme to guarantee that any two nodes share at least one pairwise key with a certain probability while [11] proposed schemes to find multiple pairwise keys between nodes. Key pre-distribution schemes utilizing location information were introduced in [12]. Other WSN security works include authentication [13], Denial-of-Service (DOS) attacks [14], routing security [15], group security [16, 17], multiple-key management [18, 19], and simple system-level security analysis [20-23].

One of the common drawbacks of those sensor network security schemes is that they do not integrate security with energy-efficient hierarchical routing architectures. Because the sensors may only want to report data to the nearby sensors, it will cause much overhead if we build secure links between any two nodes. Typically, to reduce routing overhead, a WSN should be able to self-organize itself into a “cluster” architecture [24] after sensor deployment. A “cluster” includes a group of neighboring nodes where one of the group nodes is selected as ClusterHead (CH). The “clustering” algorithms use parameters such as sensor energy level, mobility, location, etc. to form clusters and determine CHs. Our work reported here assumes that an energy-efficient clustering algorithm called “load balancing clustering” [24] is used to form clusters among sensors. Data is aggregated by the CH that removes duplicated or redundant information. The aggregation can also be achieved by having nodes closer to the CH process the data coming from nodes farther away through eavesdropping.

Most of traditional sensor network **security** schemes [9, 12] just focus on *end-to-end* security issues and ignore WSN routing details. They do not consider the low-energy routing architecture and simply assume the entire network uses tree- or flat- based topology. It is necessary and beneficial to take cluster-based communication architecture [2, 3] into consideration to reduce key management overhead in secure WSNs. In this research, we integrate WSN security issues with a cluster-based routing architecture with enhanced reliability and scalability. We call our scheme **SPECTRA** (*Secure Power-Efficient Clustered-Topology Routing Algorithm*). Our scheme integrates security with WSN topology discovery and routing procedure. Our results shows that the proposed clustering-based keying/re-keying scheme can significantly save energy compared to those works based on general flat routing topology (see Section VII) [1].

The contributions and innovations of our proposed WSN security scheme include the following four aspects:

(1) Seamless integration of security with scalable WSN routing protocols: Existing work overlooks the idea that security scheme should be seamlessly integrated with the special characteristics of WSN architecture, especially routing protocols; otherwise, the security scheme may not be practical or energy-efficient from the network protocol point of view [3]. In particular, most of the existing WSN security strategies focus only on key management / security algorithms. For example, all existing key-predistribution schemes try to establish pairwise keys between each pair of nodes. However, most sensors do not need to establish a direct point-to-point secure channel with sensors multiple hops away since WSNs use hop-to-hop communication techniques to achieve long distance transmission. The scheme in SPINS [9] simply assumes a flooding-based, spanning-tree architecture with the BS as the tree-root. However, the establishment and maintenance of a global spanning tree in a large-scale WSN

[1] Energy consumption is the top concern in tiny, battery-driven sensor network [3].

with a large footprint may not only bring unacceptable communication overhead and thus increased energy consumption [2], but also cause a large transmission delay, which also make assumption of time synchronization in μTESLA [9] (a broadcast authentication protocol) impractical.

On the contrary, we design our security scheme with considerations of WSN hierarchical routing through a *multiple-level keying*. Our scheme is highly practical because it is designed to integrate routing layer and security protocol without sacrificing power. It is a dynamic, distributed protocol where security is provided independent of central control. Another important feature of our work is that it has a robust hop-to-hop transmission scheme and can recover from multiple key losses.

(2) Dynamic Security through robust re-keying: Dynamic network topology is native to WSNs because nodes can fail or be added. In the case where nodes fall out, these nodes must be removed from the network completely. In the case of node addition, a protocol must be able to distinguish between legitimate node addition and attempted enemy infiltration. Given these reasons, we present adaptive security scheme for WSN applications in order to adapt to dynamic WSN topology. On the other hand, from time to time, network enemies can compromise sensors and all security information in those sensors may be exposed. Therefore, after key-predistribution and sensor deployment, a re-keying scheme should be used to update all types of keys. In this work, a re-keying scheme that can adapt to sensor compromise is planned.

(3) Low-complex implementation. Our work uses a symmetric-key-based scheme instead of asymmetric keying since memory usage and energy consumption are two major concerns in sensor network [3]. Asymmetric keying schemes need more complex cryptography calculations and protocols, which can bring more communication overhead compared to symmetric-key-based schemes. Our security scheme has low transmission energy due to its cluster-based key management. Because WSNs can consist of hundreds, if not thousands, of nodes and network topology/densities can change frequently, a centralized or flooding-based security scheme cannot scale well. Thus distributed algorithms and localized coordination to achieve global convergence are preferred [3].

The rest of this paper is organized as follows. Section II introduces the different type of keys and protocol messages to be used in SPECTRA. In Section III, we show the initial system setup procedure in SPECTRA. Following this setup phase, Section IV further describes the normal system operation procedure. Section V provides security functions details such as authentication and confidentiality while Section VI discusses SPECTRA capability to adapt to the dynamics of WSN topology. In Section VII, we provide performance test results of SPECTRA. Finally, Section VIII concludes this paper.

II. SPECTRA elements: security keys and protocol messages

A. Security Keys

This section provides an overview of all type of keys used within SPECTRA: the *personal key*, the *cluster key*, the *initial key*, and the *system key*. These keys are used to encrypt every message passed within the SPECTRA network. All keys within the SPECTRA network are computed through one-way radix hash functions and a pseudo-random number generator (see Section V). The pseudo random number generator is used to generate a number for the desired key length. A one-way radix hash function is then applied to this number in order to generate the key. In the case of refreshing a current key, the current key is used in place of the generated number, and the hash function is applied to the current key to generate a new key. The *personal key*s are generated prior to deployment and are stored within the memory. **Figure 1** shows the notations to be used later.

[2] Most of the sensor's power is consumed by communication, not instruction processing [3]. The energy used to communicate one bit of data can be used to execute over 1000 local instructions [3]. High energy consumption can drain sensors quickly and thus shorten the network lifetime.

Notation	Definition
$\underset{\substack{one-way \\ \rightarrow}}{f}$	One-way radix hash function used to generate the keys
$PRNG$	Pseudo Random Number Generator used to generate the number that is the Desired key length
x	The key length
K_p	*Personal Key*
$K_{Cluster}$	*Cluster Key*
$K_{current\ K_{cluster}}$	Current *cluster key* that is used to generate the refreshed key
$K_{refreshed\ \ Cluster}$	Refreshed *cluster key*
K_{System}	*System key*
$K_{current\ \ K_{System}}$	Current *system key* used to generate the refreshed key
$K_{refreshed\ \ System}$	Refreshed *system key*

Figure 1: Symbol Definitions for Key Expressions

The following expressions depict the key generation principle for the keys in **Figure 1**:

$$K_p = \underset{\substack{one-way \\ \rightarrow}}{f}\left(PRNG\ \left(x\right)\right)$$

Equation 1: Expression for generating *Personal key*s

$$K_{Cluster} = \underset{\substack{one-way \\ \rightarrow}}{f}\left(PRNG\ \left(x\right)\right)$$

Equation 2: Expression for generating *Cluster key*s

$$K_{refreshed\ \ Cluster} = \underset{\substack{one-way \\ \rightarrow}}{f}\left(PRNG\left(K_{current\ \ K_{cluster}}\right)\right)$$

Equation 3: Expression for generating refreshed *Cluster key*s

$$K_{System} = \underset{\substack{one-way \\ \rightarrow}}{f}\left(PRNG\ \left(x\right)\right)$$

Equation 4: Expression for generating *System key*s

$$K_{refreshed\ \ System} = \underset{\substack{one-way \\ \rightarrow}}{f}\left(PRNG\left(K_{current\ \ K_{System}}\right)\right)$$

Equation 5: Expression for generating refreshed *Cluster key*s

A.1 Personal key:

*Personal key*s are used for initial authentication in SPECTRA. They are pair-wise keys that are only used once for authentication and are considered invalid thereafter. This is done in order to ensure that a failed node cannot be reinserted into the network by an enemy.

(1) *Personal keys* used by **Clusterhead (CH):**

CHs use their *personal key* during *initial system setup* when first authenticating a CH with the Base-Station (BS). During system startup, every CH send a request to the BS, encrypted with their *personal key*s. The request is for obtaining a *cluster key*, and to register itself as a CH. The BS responds by sending both the latest *system key* and a new *cluster-key* encrypted with the personal-key of the requesting CH.

Before the new CH is selected, its *personal key* and unique node ID are registered with the BS. The new CH sends a request for a new cluster-key encrypted with its *personal key*, and the BS replies by sending the *system key* and *cluster-key*, encrypted with the *personal key* and *initial key* (Section V). This also serves to authenticate the new CHs.

(2) *Personal keys* used by other nodes (i.e. **non-CHs**):

Each node has a pair-wise *personal key* shared between itself and the BS. These *personal key*s are programmed into each node before system deployment, and are used to authenticate new sensor nodes joining an existing SPECTRA network. The *personal key* is used to encrypt a message broadcast when the node attempts to join the system. This message is received by the BS, which shares the pair-wise *personal key* that was used to encrypt the message. The BS then sends out the latest *system key* to the joining node, encrypted again with the *personal key*.

Once the new node has the latest *system key*, it is said to be authenticated to the network. Encrypted with this new *system key*, the node then broadcasts a request to join a cluster. All the nearby CHs respond with an advertisement, encrypted with the *system key*. This advertisement contains the cluster ID number. The node then requests to join a cluster based on the ***received signal strengths (RSS)*** of these advertisements. The CH responds by sending its *cluster key* to the joining node, encrypted with the system-key. Finally, the CH adds the new node to its node table (a table with the sensor IDs of all registered cluster members), and notifies the BS of this addition.

A.2 Cluster key

A *cluster key* is negotiated between the CH and the BS, and is then distributed to every member of the cluster. In order to get a cluster-key, the CH must authenticate itself with the BS using its *personal key*. Within a cluster, all messages are encrypted with the *cluster key*. The notion of a *cluster key* allows nodes to eavesdrop on packets in multi-hop routes to the CH. If a node receives and decrypts a data packet that duplicates its own, it will forward the original packet, and will not send the duplicated one.

The *cluster key* is the primary means of insuring data confidentiality. First, sensor data from the nodes is encrypted with the *cluster key* before being forwarded to the CH. Then a second layer of protection is achieved by encrypting the “routing header” and the already “encrypted sensor data” with the *system key*. Both keys (i.e. *cluster key* and *system key)* are needed in order to understand the contents of the message.

SPECTRA supports both *periodic* refreshing of the *cluster key* at the CH as well as refreshing in response to *compromise events*. The desired functionality is dependent on the level of security threat in the surrounding network. (1) If *periodic* refreshing is chosen, the CH generates a new *cluster key* and broadcasts it to its cluster, encrypted with the latest *system key*. (2) In the case of a *compromise event*, the CH negotiates with the BS that generates an entirely new *cluster key* for this cluster. All nodes are authenticated with the BS before receiving the new *cluster key*.

A.3 *Initial Key*

During *initial system setup* phase (i.e. just after sensor deployment) (see Section III for details), every node and CH must authenticate themselves with the BS using an *initial key* (same for all nodes), their *personal key*, and their node ID number. The purpose of the *initial key* is to encrypt the routing information of the initial authentication message. This ensures that all information within the network is encrypted and confidential. The consequence of authentication is that the BS distributes the latest *system key* which is used to replace the *initial key*. The *initial key* can thus only be used once in the lifetime of a node.

A.4. System key

In SPECTRA, all of the routing headers of all packets in the system are encrypted with the *system key* (except for the initial *authentication request*). *System key*s expire periodically in time intervals denoted as epochs. At the start of every epoch, the BS broadcasts a new *system key* three times in rapid succession[3], encrypted with the previous *system key*. All nodes in the system receive this broadcast, and use the old *system key* to decrypt the new *system key*.

(1) For Clusterhead (CH), it uses the system key for secure multi-hop routing to the BS. CHs use the *system key* for encrypting both the routing header and the data portion of a message. They also use the *system key* for authentication.

(2) For other nodes (i.e. non-CHs), they use the *system key* for authentication and encrypting *routing headers*, whereas they use both the system key and the *cluster key* for encrypting the data portions of their messages. Together, the *system key* and the *cluster key* achieve confidentiality.

B. **Protocol messages**

In SPECTRA protocol, we use three types of messages: setup messages, data messages, and system messages. *Setup messages* are used for tasks that pertain to setting up the system, such as: node and CH addition, authentication, key refreshing. *Data messages* are used for sending and receiving data as well as generating data queries. *System messages* are generally used for tasks that affect the system or its topology such as node removal. A breakdown of all the messages within our security protocol can be seen in **Figure 2**. Every message passes through the routing and security layer, where it gets a routing header that includes: (1) Message Source; (2) Message Destination; (3) Number of Hops; (4) List of Hop IDs.

The *setup messages* are used for the formation of clusters, initial and post-deployment authentication during the *system setup phase* and for system expansion (when nodes or CHs are added after initial deployment). As an example, we briefly describe a type of *setup message*, called "*Refresh System Key*". Periodically, the BS sends out new *system key*s. A message containing the refreshing *system key* should be encrypted with the last *system key*. This means that any node must have the previous *system key* in order to get the latest *system key*. This message is broadcasted from the BS to the entire network and each individual node authenticates itself by decrypting the message to get the latest *system key*.

The purpose of *Data Messages* is to collect and transmit sensor data. Nodes generate data from their sensors, encrypt it with their *cluster key*, and forward it to the CH. The CH is responsible for data collection, as well as forwarding the aggregated data to the BS. The actual contents of a data message are the data value and the node ID from which the data originated. An example of *Data Message* is "*CH-to-BS*" through which the CHs collect and aggregate data from the nodes in their cluster, and

[3] The reason of broadcasting 3 times instead of only once is to overcome the packet loss in some nodes due to wireless interference, fading, etc.

periodically send all of this data to the BS. This packet contains multiple data messages and the cluster ID. If this message was in response to a data query, then it also contains the query ID number. This data aggregation is encrypted with the *system key* and can be forwarded to the BS along a multi-hop route of CHs.

Data messages also include *routing messages*. In order to get a valid route to a given destination, such as the BS, a route must be established. Routes are established on a reactive basis when needed, usually during the *initial system setup phase*. Here we specifically mention two types of Routing messages that are important in terms of establishing secure routing table:

(1) "*RREQ*" message: When a node needs to communicate with a node for which it does not have an entry in its routing table, that node sends a Route Request message (*RREQ*) to all of its neighbors. The neighbors then forward the *RREQ* to their neighbors, each appending its node ID. This process continues until the *RREQ* is received by the destination of some node has valid routing information to the destination, which replies to the first *RREQ* message received.

(2) "*RREP*" message: Route Reply message (*RREP*) is generated in response to an *RREQ*, by the destination node, or a node that knows a route to the destination. An *RREP* contains the complete route from source to destination. This message then follows the path that it specifies back to the requesting node (the source). Unlike *RREQ*, *RREP* messages are eavesdropped by all nodes that receive them and the route in the *RREP* is extracted in order to fill in the intervening nodes routing table. This eavesdropping takes place to reduce unnecessary routing communication.

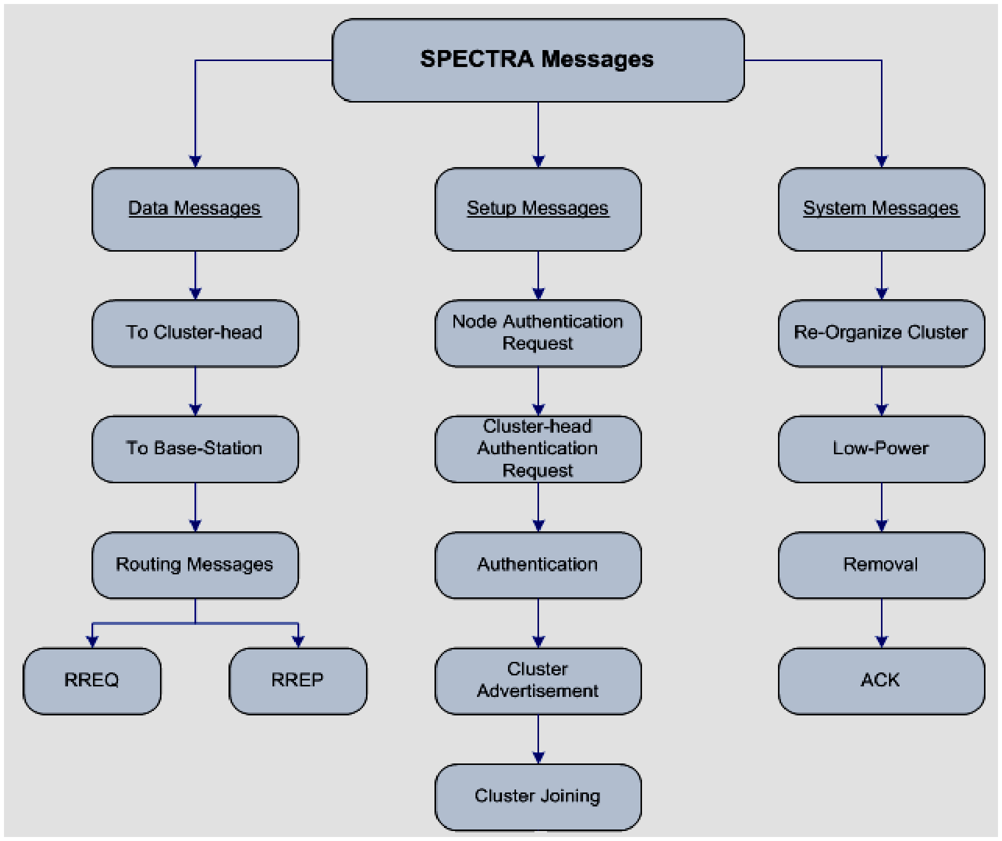


Figure 2: Messages exchanged between nodes

III. SPECTRA procedure (I): "initial system setup" phase

Before we describe the SPECTRA procedure, we briefly mention the features of the BS in a WSN: a BS collects results from all sensors [3]. Current WSN security research assumes the BS to be invulnerable to compromise, and is always trusted [9]. The BS is also assumed to have effectively unlimited battery power (such as a wall outlet), wireless transmission range, memory space, and computational capacity.

The *initial system setup* of SPECTRA consists of three phases: the *authentication phase*, *cluster organization phase*, and *route establishment*. Each of these *system setup* phases builds upon the previous phase and must be completed before the following phase can commence. A detailed description of each phase is described in the following sections.

A. Authentication Phase

In order for any node or CH to participate in the SPECTRA network, it must be authenticated. A diagram depicting an overview of the a*uthentication phase* is shown in **Figure 3**. A node is authenticated by having the latest *system key*. In order to get the latest *system key*, a node sends a request to the BS, encrypted with that node's *personal key* and the *initial key*. The BS knows that the node is authentic because the node has the *personal key* associated with its node ID. The BS replies to the node with the latest *system key*, encrypted by the *initial key* and the *personal key* of the requesting node. The node receives and decrypts the *system key*, and attempt to join a cluster.

During the *initial system setup* phase, some nodes are selected as CHs based on the load-balanced clustering algorithm [24]. When initially requesting the latest *system key*, CHs authenticate themselves in the same fashion as nodes through their *personal key* and the *initial key*. A CH needs to request a *cluster key* which they will use it to securely organize a cluster among neighboring sensors. CHs receive both the latest *system key* and a *cluster key* from the BS in reply to their *authentication request* (**Figure 4**). After this initial authentication, and for the rest of its lifetime, a node will continuously receive and decrypt the latest *system key*. This ensures *continuous authentication* (Section V).

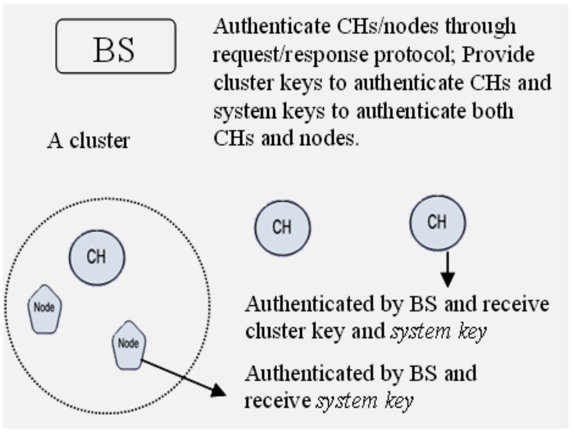

BS
Authenticate CHs/nodes through request/response protocol; Provide cluster keys to authenticate CHs and system keys to authenticate both CHs and nodes.
A cluster
CH
Node
Node
CH
CH
Authenticated by BS and receive cluster key and system key
Authenticated by BS and receive system key

Figure 3: Overview of *Authentication Phase*

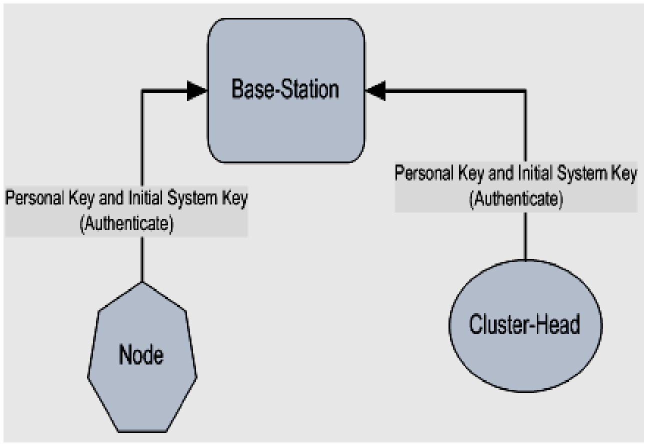

Base-Station
Personal Key and Initial System Key (Authenticate)
Personal Key and Initial System Key (Authenticate)
Node
Cluster-Head

Figure 4: CH and nodes requesting *system key*

B. Cluster Organization Phase

The *cluster organization phase* of SPECTRA sets up the network topology through the creation of clusters by using an existing energy efficient clustering algorithm which incorporates load balancing among clusters [24]. The big picture of this phase is shown in **Figure 5**.

Once a CH is selected and authenticated, it broadcasts an advertisement, encrypted with the latest *system key*. Advertisement contains the cluster ID number, and the *cluster key*. Nodes listen to these advertisements and record their Received Signal Strength (RSS). The strongest recorded RSS is associated with the nearest CH [5], and the node sends a cluster join message to this CH, encrypted with the *cluster key*. The *cluster key* is received through the *cluster advertisement*.

As *cluster joining* requests are received, the CH adds those nodes to its cluster member registry. The CH keeps a counter that is reset whenever a node joins its cluster. When the counter expires, the CH sends a cluster organization report to the BS, encrypted with the *system key*. The cluster organization report is complete with the cluster ID, the CH ID, the current *cluster key*, and the *cluster member registry*.

The *cluster member registry* is a list of all nodes within a given cluster. The BS keeps track of network topology through the *cluster member registry* table in each CH. Whenever there is a change in the topology of a cluster, a new cluster organization report is sent to the BS. This knowledge is used in the event of CH compromise in order to re-organize the cluster.

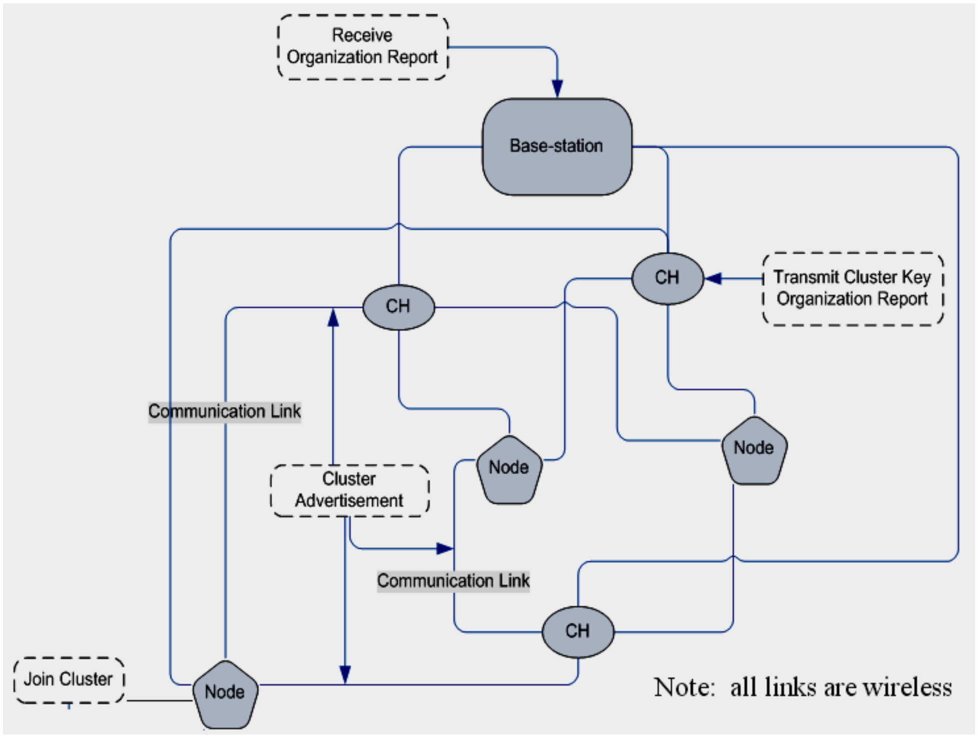


Figure 5: Overview of *Cluster Organization Phase*

C. Route Establishment

The phase of *Route Establishment* is responsible for setting up the communication routes for inter-cluster and intra-cluster routing. A diagram depicting an overview of the *route establishment phase* is shown in **Figure 6**. After clusters are organized, but before the CH sends its first cluster organization report, CHs find a route to the BS. In specific, if the BS is not one of its neighbors then the CH broadcasts a Route Request (*RREQ*, see **Figure 7**) message. A neighbor is defined to be a node who's RSS is above a certain threshold, and every hop of route must occur between neighbors.

All routing messages are broadcast to neighboring CHs in a cluster-to-cluster way and CHs keep track of the sequence number of each message. When a CH receives a *RREQ* message, it checks to see if the requested destination is one of its neighbors. (1) If the current recipient is NOT a neighbor of the requested destination, then it forwards the *RREQ* to all of its neighbors through a broadcast encrypted with the *system key*. It appends its own node ID to the route contained within the *RREQ* before forwarding the message. (2) If the receiving node is the destination of the *RREQ*, then it generates a Route Reply (*RREP*) message containing the whole route from source to destination (**Figure 7**). Only the first received *RREQ* is replied to and all following *RREQ* messages with the same sequence number are ignored. In the event that the *RREQ* is intended for one of the neighbors of the current recipient, the modified *RREQ* is forwarded only to the destination.

This route discovering process is used for both CH to BS routing, and for node to CH routing within a cluster. Like *RREQ* messages, *RREP* messages are encrypted with the *system key*, allowing both nodes and CHs to eavesdrop on routing information (**Figure 8**). This allows them to fill in their own routing tables without sending additional *RREQ* messages.

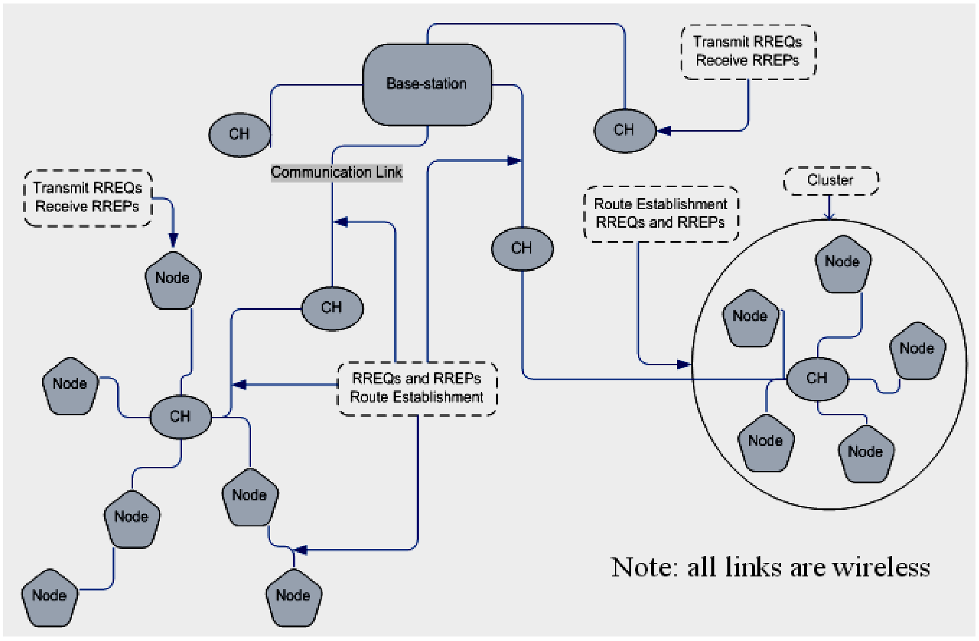


Figure 6: Overview of *Route Establishment Phase*

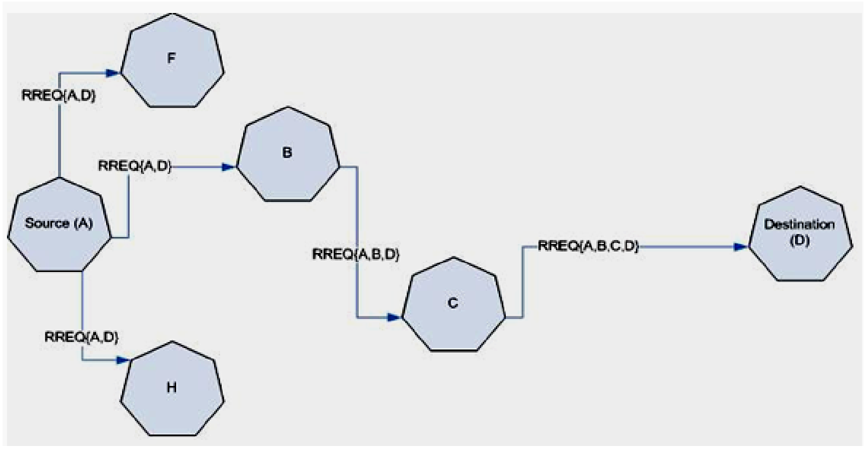


Figure 7: *RREQ* propagating through network to Destination node

VI. SPECTRA protocol procedure (ii): "system operation phase"

The *system operation phase* of the SPECTRA network commences after the *initial system setup phase*. The regular activities of a SPECTRA network are primarily concerned with data transmission and continuous authentication.

A. Data Collection, Aggregation, and Transmission

All data is generated at the sensor node after an event or a certain time period. When a node has new data, the data is forwarded to the CH, where it is stored and aggregated. The data then follows a direct or multi-hop route to the destination as shown in **Figure 9**. In particular, the CH maintains a collection of the most recent data generated by each of the member nodes in its cluster. This data is aggregated to avoid duplication. Data duplication is eliminated by truncating sample values that are similar and come from the same region of the sensor network. The specific data aggregation policy is implemented at the application layer of the CH. SPECTRA does not contain any native data aggregation policy, but supports any proper data aggregation policy. Third party data aggregation algorithms, such as the CAG algorithm [31] can be easily integrated with the SPECTRA.

When aggregated data is sent to the BS, it is encrypted only with the *system key*. This allows other CHs along the route of the SPECTRA network to perform further data aggregation.

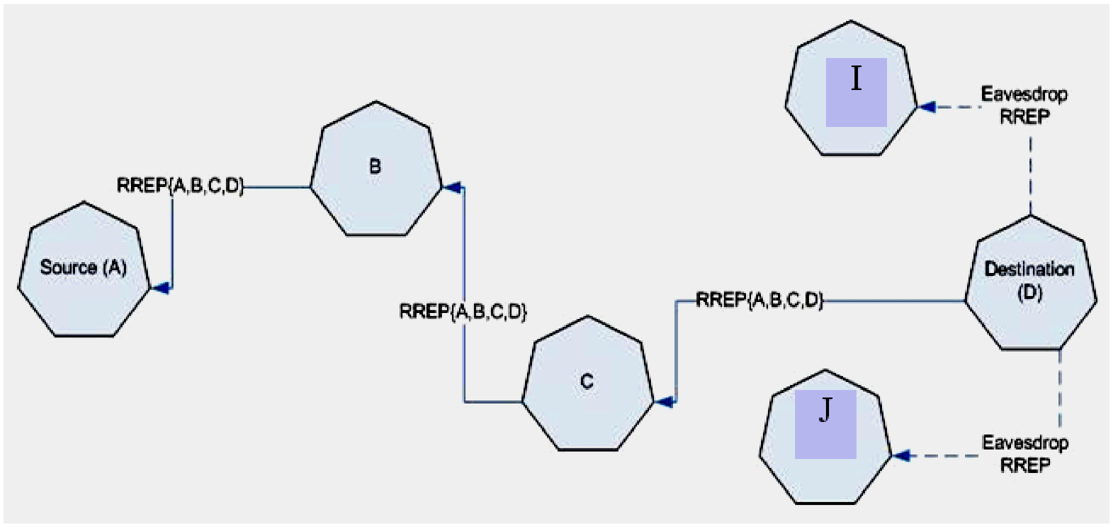


Figure 8: *RREP* generated at Destination and sent back to Source, eavesdrop by surrounding nodes

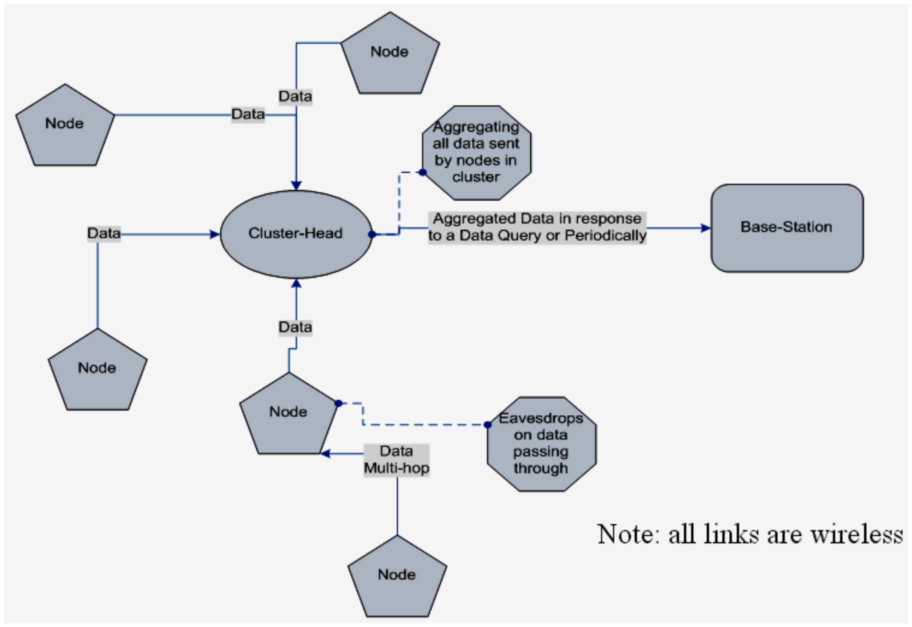

Node
Data
Node
Data
Aggregating all data sent by nodes in cluster
Data
Cluster-Head
Aggregated Data in response to a Data Query or Periodically
Base-Station
Data
Node
Node
Data Multi-hop
Eavesdrops on data passing through
Node
Note: all links are wireless

Figure 9: Single-hop and multi-hop data aggregation

B. Continuous Authentication

All nodes and CHs in a SPECTRA network are continuously and periodically authenticated. This authentication is achieved through the periodic refreshing of the *system key*. A *system key* is only valid for a period of time that is referred to as an epoch. At the start of every epoch the latest *system key* is broadcast three times in rapid succession by the BS, encrypted with the previous *system key* (Equation 6 shows the expression for a refreshed *system key*). The *system key* is broadcast three times in order to reduce the effects of wireless errors and the resulting chances of a node failing to authenticate.

$$\underset{\substack{one-way \\ \rightarrow}}{f}\left(PRNG\left(K_{current\ K_{System}}\right)\right)=K_{refreshed\ System}$$

Equation 6: Expression for refreshed *system key* (See *Figure 1* for symbol definition)

Any message whose routing header is not encrypted with the latest *system key* is disregarded and not *ACK*ed. Therefore, a node will be totally ignored by the rest of the SPECTRA network if the node does not have the latest *system key*. This guarantees that a node tampered by the enemy does not have the latest *system key* when it attempts to rejoin the network. The node cannot attempt to rejoin the SPECTRA network, because each *personal key* can only be used once to acquire the latest *system key* (this is generally done during *system setup*).

C. Eavesdropping to reduce redundancy

Eavesdropping occurs in several parts of a SPECTRA network to reduce unnecessary communication overhead. Data messages within a cluster are eavesdropped by any nodes that they pass through on their way to the CH. Since all data messages within a cluster are encrypted with the *cluster key*, nodes can decrypt and examine the contents of all data messages that pass through them. This allows nodes to avoid sending data messages regarding to duplicating events that other nodes in the cluster have already reported. Similarly, data aggregation can also be performed by all intervening CHs when messages pass along the CHs to the BS. Like all routing messages, *RREP* routing messages are encrypted with the *system key*, which allows intermediate nodes to append the corresponding route information in RREP to their routing tables.

D. Summary on the roles of each WSN component

To better understand the entire picture of all phases, we briefly summarize the functions of each component (CHs, nodes, clusters) in SPECTRA as follows.

Role of Base-station: The BS issues *cluster key*s to authenticated CHs during the *cluster organization phase*. It is also responsible for periodically broadcasting the latest *system key* to the network. The functions of the BS can be summarized as follows:

- Data querying (requesting data from the network)
- Collect information regarding to network topology from all CHs
- Initial authentication of nodes and CHs during *initial system setup*
- Global authentication of a system once it has been set up
- Authentication of new nodes added after initial setup

Role of Clusters: Clusters are formed by using an energy efficient clustering algorithm that incorporates the load balancing among clusters [24]. Clusters are beneficial in WSNs because they reduce the distance of communication required for nodes within the network [3, 24], i.e. every node does not have to directly communicate with the BS, increasing system lifetime. Cluster formation begins with an authenticated CH that requests a *cluster key* from the BS. The CH receives the new *cluster key* and decrypts it with the *system key*. After the cluster is formed, the data portions of all messages sent within the cluster are encrypted with the *cluster key*. Like all messages in the SPECTRA network, the *routing headers* of messages sent within the cluster are all encrypted with the *system key*. This *dual-key* system insures both authentication, and confidentiality. In fact, the *system key* authenticates a node as a member of the network, whereas the *cluster key* authenticates a node as a member of the cluster.

Role of Clusterhead in SPECTRA: CHs are nodes with additional responsibilities. CH's main function within the cluster is to aggregate data received from cluster member nodes. Aggregated data can be forwarded to the BS in response to a data query through the wireless routing backbone formed by all CHs. During *system setup*, CHs organize nearby nodes into a cluster. They are also responsible for acquiring and issuing the cluster-key to the nodes in their cluster. Each CH obtains a *cluster key* from the BS after the *initial system setup phase*. Then the CH transmits the *cluster key* to nearby nodes in response to their requests to join the cluster, encrypted with the *system key*.

V. SPECTRA: security management

SPECTRA is innovative in its use of multiple keys for encrypting each message. This makes node compromise and key compromise extremely difficult. To intercept a message, not only must the right keys be known, but it must also be known in which order to apply them to a given message. This section goes through the detailed encryption and dual keys scheme and discuss how the system responds to compromised nodes or CHs.

A. Encryption Approach

The SPECTRA network utilizes multiple keys to achieve security, authentication, and confidentiality. Due to the limitations of sensor nodes, all keys within SPECTRA are symmetric. The symmetric keys are simpler, smaller, and computationally less intensive than asymmetric keys. As mentioned earlier, SPECTRA uses three main keys: the *system key*, the *personal key* and the *cluster key*. The *system key* is used for global authentication purposes and is periodically refreshed. The *personal key* is used for initial node authentication during the *system setup phase*. The *cluster key* is used for security within a cluster, and is used to encrypt the data portions of all messages exchanged within the cluster.

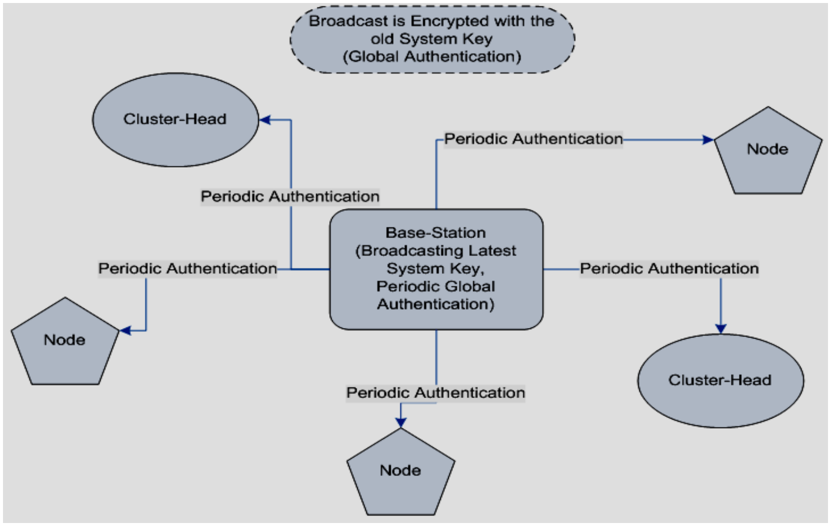


Figure 10: One of three broadcasts done during Global Authentication

Encryption in SPECTRA is achieved through one-way radix hash functions which have features such as computational ease, low memory and resource overhead. Note that SPECTRA is not limited to the hash-function-based encryption algorithm and was designed to support various methods. The choice of encryption is dependent on the application and the network environment (harsh environments call for superior encryption algorithms). The one-way radix transformation hash function implemented in SPECTRA utilizes a digital value that allows it to change base (or radix). Then, a decimal or octal key can be transformed into a hexadecimal key (hash value). This creates an infeasible reverse algorithm because the original base and the final base are different [26].

B. Authentication/ Confidentiality

During the *initial system setup phase*, SPECTRA achieves authentication through each node using its *personal key* and *initial key*. Once the *initial system setup phase* has completed, SPECTRA authenticates the entire system by periodically refreshing the *system key*. A node's or CH's *personal key* must be used in order to get the *system key* for the first time. The global authentication is achieved by periodically refreshing the *system key*. Similar to the well-known SPINS [9], this feature allows every entity in the network to be confirmed and reduces the chances of compromise because compromise has to occur before the system is authenticated again [9]. In SPECTRA, the *system key* is broadcast three times in rapid succession to the network at the beginning of every epoch. **Figure 10** shows how SPECTRA periodically refresh the network with latest *system key*.

An *epoch* is defined to be a period of time that is less than the predicted time required for node compromise. The epoch time is dependent on the network environment and is determined by an environment analysis. At the start of every epoch, the *system key* is broadcast three times in rapid succession to overcome transmission failure from wireless fading, and packet error rates [5].

Similar to the above authentication procedure, SPECTRA achieves confidentiality through the use of keys and encryption scheme. In situations when SPECTRA uses two different keys to encrypt a message, the node needs to have knowledge of both types of keys and the order to use them, which enhances the confidentiality and node identity.

C. Multiple Keys

Multiple keys are important in the SPECTRA network because they make compromise exceptionally difficult and provide two levels of authentication. Not only must a compromised node have knowledge of three different keys (i.e., personal key, cluster key and system key), but also know exactly when to use them. Also, because of different keys and message sizes, it is extremely difficult to decipher the different portions of the message.

SPECTRA uses two keys to provide confidentiality and authentication at every step in the network. All routing information of any message passed within the SPECTRA network is encrypted with the *system key* (or the *initial key* during the *initial system setup phase*, see section III) while the data portion is encrypted by the *system key* and the *cluster key*, or the *personal key* of the node. Therefore, if a node is lacking of the *system key*, no information can be sent or received (in other words, routing cannot be understood). This provides first level authentication of the node.

The data portion of all messages within the SPECTRA network is encrypted with different kinds of keys. Correspondingly, a node needs to have knowledge of network topology and understand network functionality in order to use the correct key for decrypting the data portion. This provides second level authentication of the node.

D. Node Compromise

When one node misses the latest *system key*, it can no longer function in the system since its routing header cannot be decrypted. This node is ignored and eventually removed by the system and will be unable to reenter the SPECTRA network. Nodes that miss the *system key* due to an enemy trying to infiltrate the network by physically compromising the node will be kept out of the SPECTRA network in the same manner.

Nodes/CHs are assumed to be compromised when they fail to respond to a message that has been resent to them 5 times (fail to *ACK* 5 times). The node/CH that detects such failure broadcasts a *removal* message, encrypted with the *system key*, which notifies the SPECTRA network of the compromise event. The *removal* message results in the removal of the compromised node/CH from all routing tables of the system. CHs can be compromised or drop out due to wireless errors. Both situations are treated in the same manner and the CH is removed from the system. A compromised CH trigger the BS to inform the nodes in that cluster to re-organize and appoint a new CH.

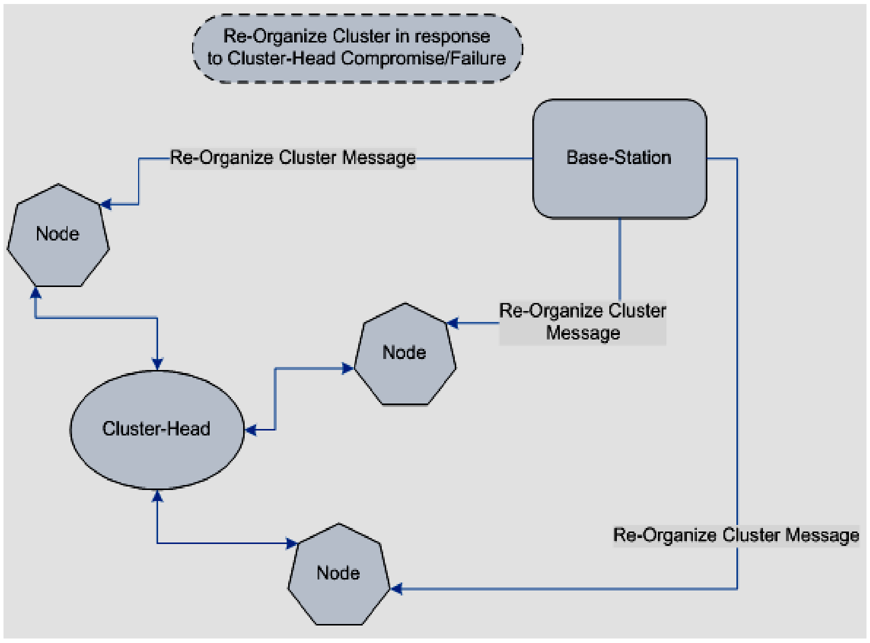

Re-Organize Cluster in response to Cluster-Head Compromise/Failure
Base-Station
Re-Organize Cluster Message
Node
Re-Organize Cluster Message
Node
Cluster-Head
Re-Organize Cluster Message
Node

Figure 11: *Re-organize cluster* Message in response to a *Removal* Message

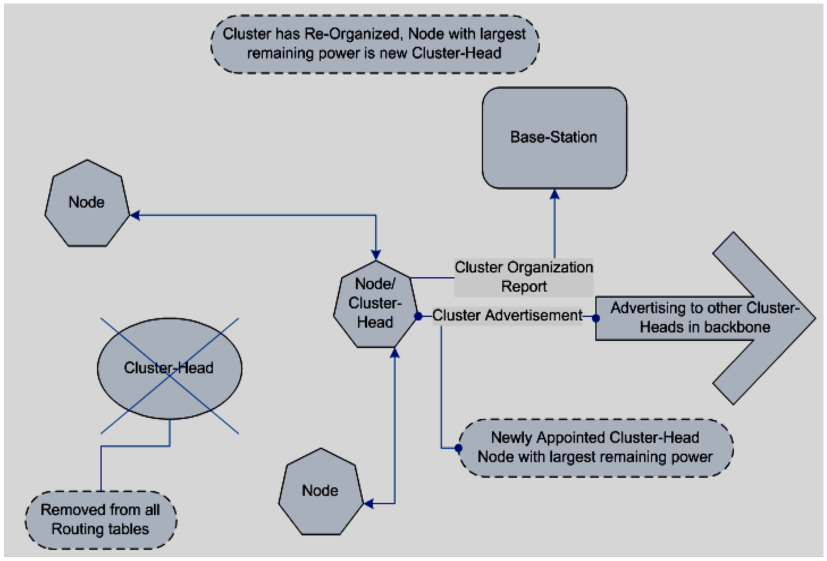

Cluster has Re-Organized, Node with largest remaining power is new Cluster-Head
Base-Station
Node
Cluster Organization Report
Node/ Cluster-Head
Cluster Advertisement
Advertising to other Cluster-Heads in backbone
Cluster-Head
Newly Appointed Cluster-Head Node with largest remaining power
Node
Removed from all Routing tables

Figure 12: Nodes have Re-Organized themselves

E. Cluster Re-Organization:

When a CH is compromised and detected, a *removal* message is broadcast to the system. The BS generates a Re-Organization message in response to this *removal* message, and sends it to the corresponding nodes as shown in **Figure 11**. This message also informs the nodes to re-organize itself and select a new CH with the largest remaining power. The newly appointed CH then broadcasts itself to the cluster backbone via a *Cluster advertisement* message (see **Figure 12**).

VI. Adaptive to dynamic WSN topology

It is imperative that a protocol should be scalable and expandable in response to the dynamic topology of WSNs. SPECTRA is designed to be robust and scaleable. As nodes fail from lack of power, system performance degrades. If no action is taken, the system will eventually cease to function. To ensure continued system functionality, new nodes and CHs must be deployed to replace the nodes and CHs that have died. Node death and the deployment of additional nodes result in a continuously changing network topology. These changes in network topology complicate the routing of messages within the network. In this section, we describe how our security scheme ensures that the security of the network is not compromised by node failure and addition.

A. Just after initial deployment

Nodes joining the SPECTRA network after the *initial system setup phase* has completed, need to authenticate themselves to acquire the latest *system key* from the BS. In specific, authentication is achieved when the node sending a message to the BS requests to join the existing network, encrypted with the node's *personal key* and *initial system key*. If the *personal key* is valid and has not been used for authentication before, the BS replies with the latest *system key*, encrypted with the requesting node's *personal key* and *initial system key*. Once the node has acquired the latest *system key*, it attempts to find and join proper cluster by broadcasting a *cluster joining* message, encrypted with the *system key*.

B. During normal operations: Node / CH Addition

After the initial setup phase is finished, some nodes may be added to compensate for the failed ones. Similar to the above case, new nodes need to undergo authentication. Once authenticated, the node broadcasts a *cluster joining* message, encrypted with the *system key*. CHs reply to the request with their cluster ID and *cluster key*, encrypted with the *system key*. The node keeps track of the RSS of all replies and joins the cluster with the highest RSS. All information regarding other clusters is removed, and only the *cluster key* from the cluster that is joined is kept. The node ID of the new node is then added to the cluster member registry of the CH which is eventually forwarded to the BS (in the cluster organization report).

During normal WSN operations, re-routing occurs periodically. Thus some nodes can be selected as new CHs. These new CHs need to securely find their cluster members. Once a new CH has been authenticated by the BS and acquired both *system key* and new *cluster key*, it proceeds to set up a cluster. The CH broadcasts a Cluster-Advertisement message that is encrypted with the *system key*. Any node that receives a stronger signal from the new CH compared to the signal from their current CH replies with a *cluster joining* message, encrypted with the *system key*. This message is broadcast and contains the node ID and the CH ID. This message is broadcast within a certain area, and therefore all nearby CHs will receive it, which enables the node's current CH to remove the node from its cluster member registry, and the new CH adds that node to its registry. Eventually the BS is notified of this change through a cluster organization report.

C. *Node / CH Death*

When a node's available power drops below a certain threshold, that node sends a Node Death message to its CH. The CH then sets a timer to three times the predicted remaining lifetime of the node. Once this timer expires the CH removes this node from its cluster member registry and broadcasts a notification to its cluster. This message instructs all nodes in the cluster to eliminate the dying node from their routing tables. CH death is handled in a similar fashion to node death. When the timer expires, the BS uses a broadcast to notify the corresponding cluster that its CH has died. This causes the nodes in the cluster to re-organize themselves and appoint the node with the largest remaining power to be the new CH.

D. "Reliable" key transmissions from CHs to the BS:

Dynamic WSN topology and wireless interference can cause frequent packet loss during re-keying operations. Some of those packets hold the latest system keys or cluster keys. We call those packets "keying packets". Thus packet loss can cause key loss. To reduce *key loss rate*, we have specifically investigated two approaches to achieving reliable *CH-to-BS* transmission:

(1) Multipath routing without packet loss recovery scheme. In this case, we establish multiple paths from a CH to the BS. Multiple copies of a keying packet are transmitted through these different paths. Thus even a path lost a keying packet, other paths can possibly deliver the copies of that packet.

(2) Single-path routing with local recovery. **Figure 13** shows our reliable *CH-to-BS* transmission scheme using a single path with local link failure repair. In Figure 13, Clusters **(G, B, D), (H, A, E), and (I, C, F)** have the same number of hops to the BS. Suppose *CH A* is not reachable due to its battery depletion or the unreliable wireless link from *B* to *A*. We first go backward one *hop* to *B* and check the reachability of *A*'s neighboring *CHs H* and *E*. If any one of them is reachable, we use it. If *H* and *E* both fail, we go back one more *hop* to *CH K*, and let *K* find out *B*'s neighboring *CHs* and try to deliver the packets. We have conducted extensive routing simulations and found out that the '*single-path with local repair'* scheme can achieve higher transmission reliability than general end-to-end retransmission scheme [32]. (In next section, we will discuss our reliability test results). One reason could be that the establishment of multiple paths from any *CH* to the *BS* needs higher maintenance overhead and generates multiple copies of the same packet, which consumes more energy (for wireless communication) than single-path approach.

Note: Clusters **(G, B, D), (H, A, E), and (I, C, F)** have the same # of hops to the BS.

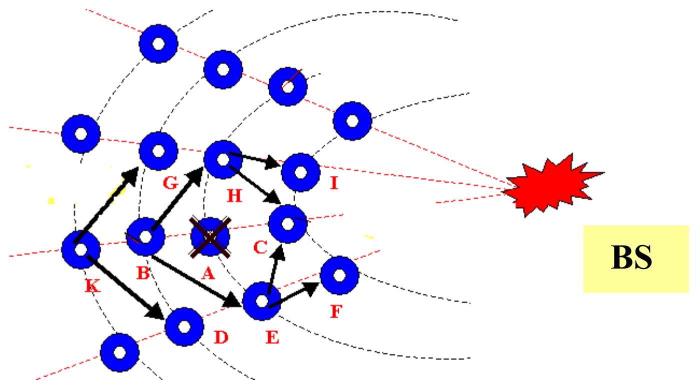


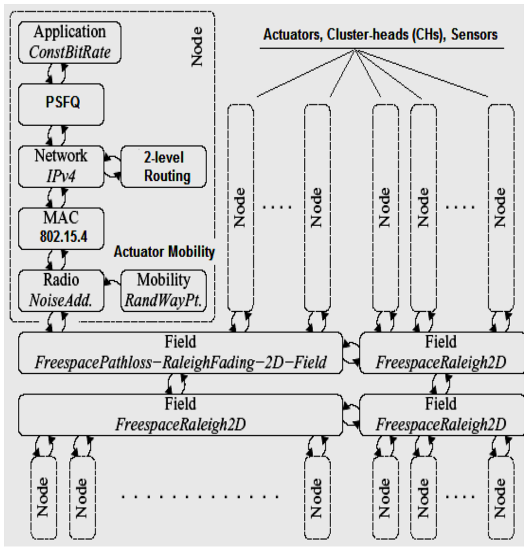
Figure 13: Key loss recovery scheme

VII. Performance Tests

A. Jist+SWANS based WSAN security simulation

The WSN security performance analysis results can be obtained through a Java-based simulation engine. The simulation engine is comprised of JiST [27] (Java in Simulation Time), and SWANS [28] (Scalable Wireless Ad-hoc Networks Simulator). SWANS can achieve high performance and scalability (e.g., >1000 nodes) from its representation of the field entity.

Figure 14 shows our revised SWANS simulator for SPECTRA performance test purpose. To detect packet loss, we have implemented a reliable "Transport Layer protocol" above the cluster-based routing protocol based on PSFQ [37]. We implement a CH-by-CH NACK (Negative ACKnowledge) based reliability scheme. A gap in the sequence number of sent packets indicates packet loss. Each CH maintains a list of missing packets. When a loss is detected, a tuple containing a source ID and sequence number of the lost packet is inserted into this list. Entries in the "missing packets" list are piggybacked in outgoing transmissions, and CHs infer losses by overhearing this transmission. CHs keep a small cache of recently transmitted packets, from which a node can repair losses reported by its last hop CH. Besides CH-by-CH loss recovery, we also implement *end-to-end* recovery. This is because those heavy packet losses can lead to large missing packet lists that might exceed the memory of the CH. Our *end-to-end* recovery scheme leverages the fact that the base station has significantly more memory and can keep track of all missing packets. The base station attempts CH-by-CH recovery of a missing packet. When one of the CHs notices that it has seen a packet loss from the corresponding source, but does not have a cached copy of that packet, it adds that recovery request to its missing packets list. This request is propagated downward in this manner (using the same mechanisms described for CH-by-CH recovery) until it reaches the source. Since the source maintains generated packets in its memory, it can repair the missing packet.



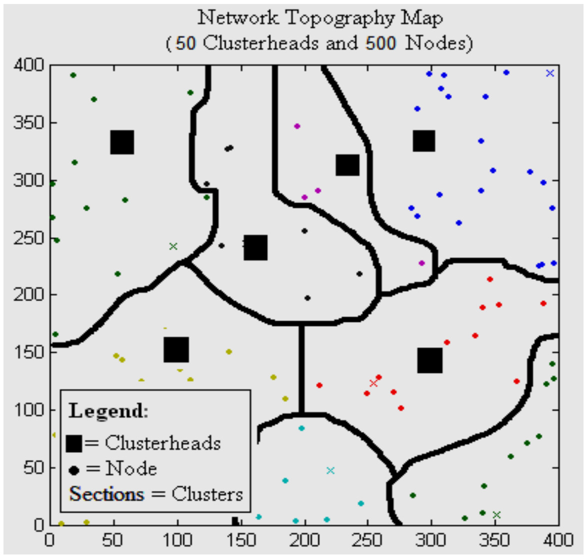


Figure 14: WSN Simulator Architecture **Figure 15**: cluster-based topology (Unit: meters)

Our simulation scenario is a WSN with 100 ~ 1000 nodes (we change the node density and the total number of sensors to measure WSN performance such as energy consumption, routing overhead, security calculation overhead, etc.). The load-balanced clustering algorithm [24] is used to select some nodes as CHs. The wireless communication range of a CH is set up to 50 meters (to achieve inter-cluster communication), while the range of a cluster node is set up to 10 meters (they only participate in intra-cluster communication). The sensors are assumed to be static. The CH/nodes are assume to have a limited initial energy (we set to 500 uW) and will fail when they run out of power. While the BS is assume to have unlimited power and can broadcast a message to the entire WSN (i.e. have unlimited radio range). **Figure 15** shows a sample of the large-scale (500 nodes) WSN simulation topology with cluster architecture we used. Its area is a 400 x 400 square. The small square black blocks represent selected CHs.

B. Energy model

SWANS does not natively have a function or layer that tracks energy consumption, which is one of the most important performance metrics. A battery layer was added in order to keep track of energy during simulation runtime. We use the same radio model as discussed in [38] which is the first order radio model. In this model, a radio dissipates Eelec = 50 nJ/bit to run the transmitter or receiver circuitry and **Î**amp = 100 pJ/bit/m^2 for the transmitter amplifier. The radios have power control and can expend the minimum required energy to reach the intended recipients. The radios can be turned off to avoid receiving unintended transmissions. An r^2 energy loss is used for channel transmission [5]. The equations used to calculate transmission costs and receiving costs for a ***k*-bit** message and a **distance *d*** are shown below:

Transmitting

$E_{Tx}(k, d) = E_{Tx-elec}(k) + E_{Tx-amp}(k,d)$

$E_{Tx}(k, d) = E_{elec} * k + \in_{amp} * k * d^2$

Receiving

$E_{Rx}(k) = E_{Rx-elec}(k)$

$E_{Rx}(k) = E_{elec} * k$

Receiving is also a high cost operation, therefore, the number of receives and transmissions should be minimal. In our simulations, we used a packet length *k* of 2000 bits. With these radio parameters, when d^2 is 500m^2, the energy spent in the amplifier part equals the energy spent in the electronics part, and therefore, the cost to transmit a packet will be twice the cost to receive.

C. SPECTRA Performance Test Results

(1) Energy Efficiency Test:

Since most of WSN nodes are tiny sensors, the security protocols should have low energy consumption. We have investigated the energy-efficiency of our SPECTRA scheme. Our scheme has the lowest energy consumption (for all nodes in a WSN) compared to the security based on other schemes such as LEACH [38], ZRP [39], and the flat-topology based routing strategy (see **Figure 16**).

(2) Reliability Test:

We adopt hop-to-hop local recovery scheme to handle packet losses (see Section VI.D). **Figure 17** shows that other loss recovery architectures that are based on ACQUIRE [40] (using cluster-to-cluster recovery) or TAG [41] (using a simple spanning tree end-to-end recovery), have a higher keying packet loss rate than SPECTRA that is based on a scalable cluster routing.

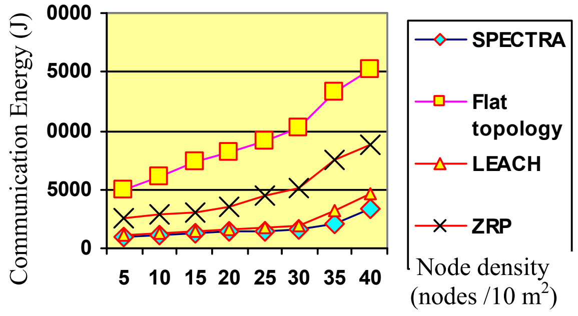


Figure 16: Energy efficiency Test

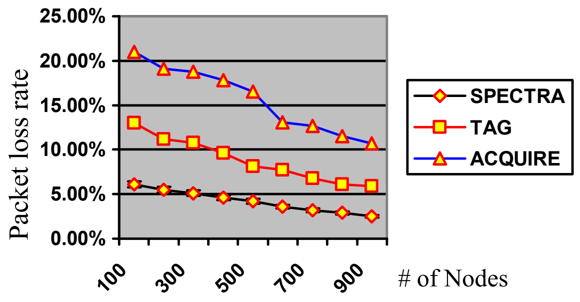


Figure 17: Reliability Test

(3) Scalability Test:

Figure 18 shows the number of chosen CHs varies with the number of sensor nodes. We can see that SPECTRA has good scalability performance. Even the network density increases dramatically, SPECTRA can still select a relatively small amount of sensors as CHs. This characteristic is very important from the viewpoint of security complexity since too many CHs can result in high inter-cluster communication overhead.

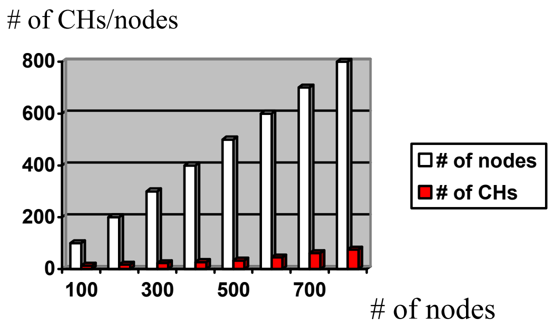


Figure 18: SPECTRA Scalability

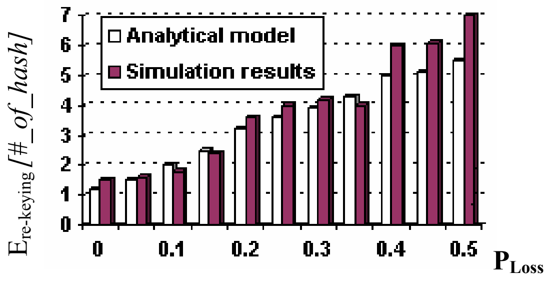


Figure 19: Security Overhead

(4) Security Overhead Analysis

We have used a first-order *Markov Chain* model to analyze the calculation/communication overhead in SPECTRA. As shown in [9], the computation of the one-way *hash function* in each sensor node consumes significant amount of energy. We therefore focus on the cost of computing hash functions during each *re-keying* session. A node may fail to receive a new *system key*, or it may receive an incorrect *system key* that cannot be authenticated by using the *hash function*. Incorrect *system keys* may also come from opponents attempting *Denial-of-service* attacks.

If the key chain buffer length is *n*, the probability of *key loss* is P_{Loss}, and the probability of *key corruption* is $P_{Corruption}$. We can derive the *expected times for hash function calculations in a re-keying cycle*, $E_{re\text{-}keying}$ *[#_of_hash],* as follows [42]:

$$E_{Re-keying}[\#\_of\_hash] = \frac{2.5 - P_0}{2 - P_0^n}\left\{ n \bullet P_0^{n-1} + (1 - P_{failure}) \bullet \sum_{i=1}^{n} (i \bullet P_0^i) \right\}$$

$$where \quad P_0 = P_{Loss} + P_{Corruption}$$

Assuming $P_{Corruption} = 0.25$, we compare the simulation and analytical results of $E_{re\text{-}keying}$ *[#_of_hash]* while varying P_{Loss} from 0.0 to 0.5. **Figure 19** indicates the validity of above analytical model [42].

VIII. Conclusions

One of the most challenging topics in WSNs is security due to the ad hoc nature, intermittent connectivity, and resource limitations. Current solutions to the security issue in WSNs were developed with only authentication and confidentiality in mind without considering energy-efficient, scalable WSN routing architecture. This is far from optimal because routing and security are closely correlated. In this paper, we have proposed a Secure Power-Efficient Clustered-Topology Routing Algorithm (SPECTRA), which integrates routing and key management to provide an energy efficient security and routing solution. SPECTRA also owns promising features such as dynamic security, robust re-keying, low-complexity, and multiple levels of encryption. Our extensive simulations and analysis have verified the practicality of the SPECTRA protocol and its scalability, reliability, and low overhead in power consumption.

Acknowledgment

The authors thank Mr. Krishna Sankar (with Cisco Inc.) for his encouragements and support during the Cisco URP project.

References

[1] C. Huang, H. Lee, and Y. Tseng, "A two-tier heterogeneous mobile Ad Hoc network architecture and its load-balance routing problem," *Mobile Networks and Applications*, Volume 9, Issue 4 (August 2004), pp. 379 - 391.

[2] I. F. Akyildiz and I. H. Kasimoglu, “Wireless Sensor and Actor Networks: Research challenges,” *Ad hoc Networks Journal* (Elsevier), 2004.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A Survey on Sensor Networks,” *IEEE Communications Magazine*, August 2002.

[4] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Microsensor Networks,” In *Proceedings of the Hawaii Conference on System Sciences*, 2000.

[5] T. S. Rappaport, *Wireless Communications*, Prentice-Hall, 1996.

[6] D. Liu and P. Ning, “Establishing Pairwise Keys in Distributed Sensor Networks,” The *10th ACM Conference on Computer and Communications Security* (CCS '03), Washington D.C., October, 2003

[7] D. Liu and P. Ning, “Location-Based Pairwise Key Establishments for Relatively Static Sensor Networks,” *ACM Workshop on Security of Ad Hoc and Sensor Networks* (SASN '03) October 31, 2003 George W. Johnson Center at George Mason University, Fairfax, VA, USA.

[8] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, “A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge,” in *Proc. Of IEEE INFOCOM'04*, March 7-11, 2004, Hongkong.

[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, “SPINS: Security Protocols for Sensor Networks,” in *Proc. of Seventh Annual International Conference on Mobile Computing and Networks* (MOBICOM 2001), July 2001.

[10] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in *Proc. of the 9th ACM conference on Computer and communications security*, Washington, D.C., USA, 2002

[11] H. Chan, A. Perrig, and D. Song, “Random Key Predistribution Schemes for Sensor Networks,” *IEEE Symposium on Research in Security and Privacy*, 2003.
[12] D. Liu and P. Ning, “Establishing Pairwise Keys in Distributed Sensor Networks,” in *Proc.of 10th ACM Conference on Computer and Communications Security* (CCS '03), Washington D.C., 2003.
[13] S. Zhu, S. Setia, S. Jajodia, and P. Ning, “An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks,” in *Proc. of IEEE Symposium on Security and Privacy (S&P'04),* Oakland, California, May 2004.
[14] A. D. Wood and J. A. Stankovic, “Denial of Service in Sensor Networks,” *IEEE Computer*, 35(10):54-62, 2002
[15] C. Karlof and D. Wagner, “Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures,” *First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.
[16] G. Wang, W. Zhang , G. Cao , and T. L. Porta, “On Supporting Distributed Collaboration in Sensor Networks,” MILCOM 2003, October, 2003.
[17] J. Zachary, “A Decentralized Approach to Secure Group Membership Testing in Distributed Sensor Networks,” MILCOM 2003, October, 2003.
[18] S. Zhu, S. Setia, and S. Jajodia, “LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks,” in *Proc. 10th ACM Conference on Computer and Communications Security* (CCS '03), Washington D.C., October, 2003.
[19] F. Hu and N. K. Sharma, “Security Considerations in Wireless Sensor Networks,” *Ad hoc Networks Journal* (accepted for publishing), *(Elsevier),* 2004.
[20] Y. Wei Law, S. Etalle, and P. H. Hartel, “Key Management with Group-Wise Pre-Deployed Keying and Secret Sharing Pre-Deployed Keying,” Centre for Telematics and Information Technology, University of Twente, The Netherlands, Technical report (TR-CTIT-02-25), July, 2002.
[21] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, and M. B. Srivastava, “On communication Security in Wireless Ad-Hoc Sensor Network,” in *Proc. Of Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE'02) June, pp. 10 - 12, 2002 Pittsburgh, Pennsylvania, USA.
[22] M. Chen, W. Cui, V. Wen, and A. Woo, “Security and Deployment Issues in a Sensor Network,” downloadable from
http://citeseer.nj.nec.com/chen00security.html.
[23] Y. W. Law, S. Dulman, S. Etalle, and P. Havinga, “Assessing Security-Critical Energy-Efficient Sensor Networks,” Department of Computer Science, University of Twente, Technical Report TR-CTIT-02-18, Jul 2002.
[24] J. Yu and P. Chong, “A Survey of Clustering Schemes for Mobile Ad-Hoc Networks.” *Network Technology Research Center.*
[25] H. Karl and A. Willig, “A Short Survey of Wireless Sensor Networks,” *TKN Technical Report TKN-03-018*, October 2003
[26] RSA Laboratories, “What is a hash function?” RSA Security Section 2.1.6. <http://www.rsasecurity.com/rsalabs/node.asp?id=2176>.
[27] R. Barr, "JiST - Java in Simulation Time Users Guide," March 19, 2004.
[28] R. Barr, "SWANS - Scalable Wireless Ad hoc Network Simulator Users Guide," March 19, 2004.
[29] D. Liu and P. Ning, “Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks,” in *Proc. of the 10th Annual Network and Distributed System Security Symposium*, pp. 263 - 276, San Diego, California. February 2003.
[30] D. Liu and P. Ning, “Multi-Level u-TESLA, A Broadcast Authentication System for Distributed Sensor Networks,” Submitted for journal publication. Also available as Technical Report, TR-2003-08, North Carolina State University, Department of Computer Science, March 2003.

[31] S. Yoon and C. Shahabi, "An Energy Conserving Clustered Aggregation Technique Leveraging Spatial Correlation," IEEE SECON poster, October 2004.

[32] F. Hu, Y. Wang, and H. Wu, "Mobile Telemedicine Sensor Networks with Low-energy Data Query and Network Lifetime Considerations," *IEEE Transactions on Mobile Computing* (To appear in the issue of the Third Quarter in 2005).

[33] F.Hu and S. Kumar, "The integration of ad hoc sensor networks and cellular networks for multi-class data transmission," *Ad Hoc Networks Journal* (Elsevier), accepted for publication in August 2004, now downloadable from Elsevier. See http://www.sciencedirect.com/science/journal/15708705 (click "articles in press").

[34] H. Harney and C. Muchenhirn, "Group Key Management Protocol (GKMP) Architecture," RFC 2094, July 1997.

[35] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch Rekeying for Secure Group Communications," in *Proc. of 10th International Word Wide Web Conference*, May 2001.

[36] T. Park and K. G. Shin, "LiSP: A Lightweight Security Protocol for Wireless Sensor Networks," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 3, August 2004.

[37] C. Y. Wan, A.T. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," *WSNA'02,* September 28, 2002, Atlanta, Georgia, USA.

[38] W. Heinzelman, "Application-Specific Protocol Architectures for Wireless Networks," PhD Thesis at the Massachusetts Institute of Technology, June 2000.

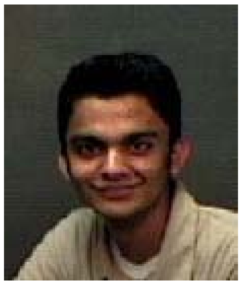
[39] Z. Haas and M. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," *IETF Internet draft for the Manet group*, 1999.

[40] N. Sadagopan, B. Krishnamachari, and A. Helmy, "The ACQUIRE mechanism for efficient querying in sensor networks," *First IEEE International Workshop on Sensor Network Protocols and Applications* (SNPA), in conjunction with IEEE ICC 2003.

[41] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," in *Proc. of the Fifth Annual Symposium on Operating Systems Design and Implementation* (OSDI), December 2002.

[42] F. Hu and C. May, "LESS: Light-wEight Security Solution for Wireless Sensor Networks Based on a Scalable Tree-Ripple-Zone Routing Scheme," *IEEE Transactions on Mobile Computing* (To appear), 2005.

Dr. Fei Hu is currently an Assistant Professor in Computer Engineering Department at RIT, NY. His research interests are in ad hoc sensor networks, 3G wireless & mobile networks and network security. He received his PhD degree in Electrical and Computer Engineering from Clarkson University in 2002. His PhD research was on high-performance transmission issues in wireless networks. He obtained his BS and MS degrees from Shanghai Tiedao University (China) in 1993 and 1996, respectively. From 1996 to 1999, Dr. Hu worked as a Senior Networking Engineer in Shanghai Networking Lab (Tiedao) and Shanghai Lucent Inc., where he completed over ten large projects on high-performance networks. Dr. Fei Hu is a Full Sigmaxi Member (an honor for scientists and engineers) and IEEE chapter officer. In past a few years he has published over 30 papers in the top networking journals and conference proceedings including IEEE Transactions on Vehicular Technology, IEEE Communication Surveys, Computer Network Journal (Elsevier), IEEE Infocom, IEEE Globecom, and IEEE ICC.



Mr. Waqaas Siddiqui is a graduate student in Computer Engineering department at RIT under the supervision of Dr. Fei Hu. His research focus is security in wireless sensor networks through a scalable routing architecture.

Dr. Xiaojun Cao received the B.S. degree from Tsinghua University, Beijing, China, in 1996, the M.S. degree from the Chinese Academy of Sciences, China, in 1999, and the PhD degree from the State University of New York at Buffalo in 2004. He is currently an Assistant Professor with the Department of Information Technology at Rochester Institute of Technology, Rochester, NY. His research interests include modeling, analysis, and protocols/algorithms design of communication networks.