WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2008; **8**:1–24
Published online 12 September 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/wcm.422

Security in wireless sensor networks

Venkata C. Giruka, Mukesh Singhal*,†, James Royalty and Srilekha Varanasi

Department of Computer Science, Laboratory for Advanced Networking, University of Kentucky, Lexington, KY 40506, U.S.A.

Summary

With sensor networks on the verge of deployment, security issues pertaining to the sensor networks are in the limelight. Though the security in sensor networks share many characteristics with wireless ad hoc networks, the two fields are rapidly diverging due to the fundamental differences between the make-up and goals of the two types of networks. Perhaps the greatest dividing difference is the energy and computational abilities. Sensor nodes are typically smaller, less powerful, and more prone to failure than nodes in an ad hoc network. These differences indicate that protocols that are valid in the context of ad-hoc networks may not be directly applicable for sensor networks. In this paper, we survey the state of art in securing wireless sensor networks. We review several protocols that provide security in sensor networks, with an emphasis on authentication, key management and distribution, secure routing, and methods for intrusion detection. Copyright © 2006 John Wiley & Sons, Ltd.

KEY WORDS: sensor networks; security; protocols

1. Introduction

Recent advances in the fabrication of small-scale computing hardware capable of performing specific tasks, along with the reduced size of wireless communication devices, has given rise to a new area of research, viz., sensor networks. A sensor network [1] is a collection of small, low-power computing devices deployed within an *environment* for the purpose of sensing (monitoring) physical *phenomena* on behalf of an *observer*. In this context, an observer is any end user(s) interested in data collected by the sensor network. An environment can be any spatial region where the sensor network is deployed such as a battlefield, a volcano, a home, or inside the human body. A phenomenon is the 'entity' that is of interest to the observer. Sensor networks are useful in environments too hostile, or too small for either human observers or larger scale wireless monitoring devices. Example, applications include monitoring of the early-childhood learning process, habitat monitoring, perimeter protection, and biomedical monitoring.

The basic building block of a sensor network is an individual sensor node. A typical node might, for example, monitor temperature, light, sound, or odor, the final choice is application dependent. A sensor node is characterized by its small size, meager computing power, low communication bandwidth, and a limited energy

*Correspondence to: Mukesh Singhal, Department of Computer Science, Laboratory for Advanced Networking, University of Kentucky, Lexington, KY 40506, U.S.A.
†E-mail: singhal@cs.uky.edu

Contract/grant sponsor: NSF; contract/grant numbers: IIS-0242384; IIS-0324836; CCR-0100040.

supply. Given these limitations, sensor nodes are typically deployed redundantly in large number (possibly on the order of thousands) within a target environment. Currently, sensor nodes exist only on a macro-scale, that is, they are visible to the naked human eye.

Limitations on size do effect how and where sensor networks can be deployed. Researches envision a micro-scale and even a nano-scale sensor nodes that could be deployed, say, within the human body or some other confined space. This means that the number of micro- and/or nano-scale sensor nodes deployed within some environment could approach (on the order) the number of hosts in today's Internet. However, currently such networks are only now being theoretically investigated. The protocols and methods presented in this paper are primarily applicable to macro scale sensor networks.

Once deployed within some environment, sensor nodes work to form a network and begin to collect and report data. Nodes communicate in an ad hoc fashion over wireless channels, and are susceptible to various forms of attack. From a high level perspective, attacks includes [2] interception of communication, subversion of one or more sensor nodes, falsification of data and/or nodes, and denial of service. These attacks differ slightly from attacks common to ad hoc networks. This is partly due to the fact that sensor nodes are "unattended" [3], that is, a user is typically not associated with a sensor node.

1.1. Comparison With Ad Hoc Networks

In contrast with nodes in traditional ad hoc networks, sensor nodes are (often) deployed in an unprotected, hostile environments and are thus vulnerable to physical attacks [4]. For instance, sensor nodes can be destroyed, not only by an attacker, but by the very phenomena they are monitoring. They can be tampered with, physically replaced with malicious impostors, and so forth. Thus, in addition to electronic security, physical security in sensor networks is vital.

A typical ad hoc network may consists of hundreds of nodes [5]. A sensor network may contain a number of nodes several orders of magnitude greater than the largest ad hoc network. This is due, in part, to the fact that sensor nodes are often deployed densely, and with a degree of redundancy in order to compensate for their high failure rate. Also, as mentioned previously, nodes in an ad hoc network are usually associated with some user(s). Thus, we could say that ad hoc networks are *user driven*; they exist to serve the needs of a group of users. In contrast, sensor nodes are unattended and the resulting network is *data driven*; it exists to collect and report data. Several sensors would be required to monitor say, a chemical process within the human body, than the number of PDA users within several groups.

Another difference from ad hoc networks is that the topology of a sensor network is expected to change very frequently, not only due to mobility as in ad hoc networks but also due to the high failure rate of sensor nodes. As a result of such frequent changes, sensor nodes often communicate via broadcast (as opposed to point-to-point in ad hoc networks) as route path discovery can be too expensive, both in terms of time and energy. Sensor nodes may be dramatically less powerful, in terms of computational resources (speed and memory) and energy (battery life), than the least powerful node in an ad hoc network. For instance, even "simple" ad hoc nodes are capable of route discovery, data storage (to some extent), encryption, and some form of power replenishment. Sensor nodes in general have little or no storage capacity or processor cycles to spare outside what is required for sensing and normal operation. Thus, little remains for expensive route discovery and/or encryption operations. In terms of energy, when an ad hoc node runs out of power (a battery dies, for example) their power source can be replenished either via connection to AC power or fresh batteries installed by a human user. Sensor nodes, in contrast, are expected to aggressively conserve energy as their power sources cannot easily be replenished; often they are not replenished at all and sensor nodes simply "die" once they run out of energy.

Finally, unlike ad hoc nodes, sensor nodes may not have global identifiers (like IP addresses) due to the potentially large number of sensors and the overhead required for such an addressing model. Put another way, individual sensor nodes are not normally addressable by an observer. If direct addressing is necessary then typically an observer refers to a collection of sensor nodes within a given area or by sensing task. This represents only topical differences between ad hoc and sensor networks. Interested readers are referred to [4] for more in-depth treatments.

1.2. General Vulnerabilities

Given this high level description of a sensor network, and ignoring communication details for now, we can delineate some general areas where the network is vulnerable to attacks [2,6,7].

Interception. As stated previously, sensor nodes communicate over wireless channels and thus, all traffic is susceptible to interception. Interception of

communication could lead to a loss of private information like passwords, security system parameters, details about the system being monitored, and so on. Knowledge of this and other information may lead to other forms of attack. The solution in traditional ad hoc networks is to encrypt communication packets. However, due to the meager computing resources of sensor nodes, strong encryption (or any encryption at all) may be impossible.

Subversion. One or more sensor nodes could be captured (electronically) but can continue as a valid member of the network. This captured node could be used to intercept communication (without listening externally, from outside the network), inject false data readings, disrupt routing, or mount other types of attacks from "inside" the network.

Falsification. A malicious intruder may add a "false" node to the network. This node could operate as a legitimate member of the sensor network and steal information, prevent the flow of communication between valid nodes, and/or inject fraudulent data into the network.

Denial of service. A malicious intruder can mount a denial of service attack using any number of well-known techniques applicable to traditional ad hoc networks. Examples include signal jamming and traffic flooding. Overloaded sensor nodes, affected by the denial of service attack, may be unable to sense the phenomenon or may acquire inaccurate readings. Moreover, a successful denial of service attack on the base station would render the entire network useless.

Physical corruption. Nodes (on the macro scale, at least) can be removed at will and analyzed off-line. This analysis could help an attacker to discover encryption keys, learn the structure of the network, and so forth. If a base station is part of the sensor network, this could represent a single point of failure for the entire network.

Apart from these attacks, there are several attacks [2,6,7] possible on sensor networks, some of which may be specific to a protocol at a particular layer in the protocol stack. For instance, an intruder may attack the authentication protocol itself, or may attack the underlying routing protocol and may bring down the network communication. Securing sensor networks against such attacks, given the meager computation and battery resources of sensor nodes, is a non-trivial task.

1.3. Outline of the Paper

The focus of the rest of the paper is primarily on electronic security in the form of protocols related to authentication, key establishment and management, routing, and intrusion detection (ID). While we do briefly discuss physical security, interested readers are referred to [8–10] for more in-depth treatments. In Section 2 we discuss issues of authentication and confidentiality. Section 3 explains key agreement and management. Section 4 is devoted to various types of attacks against secure routing protocols in sensor networks. Section 5 presents the concept of denial of service attack in sensor networks and discusses some possible defenses. Section 6 discusses various methods of ID, including a short summary of physical security issues in Section 7. We summarize review in Section 8.

2. Authentication and Confidentiality

Authentication and confidentiality of the data are critical to the security of any network. Given the meager resources of sensor nodes, authentication and confidentiality may have to be realized at a trade-off between the desired security level and the processing requirements. A common approach to provide confidentiality is to encrypt sensitive data with a secret key possessed by only the intended receivers. However, one must consider the resource constraints when choosing cryptographic primitives and security protocols in sensor networks. In this section, first we present a discussion on cryptographic algorithms and energy efficiency. Subsequent subsections present some security models that use powerful base stations, which perform most of the computational and storage tasks, to provide authentication and/or confidentiality.

2.1. Cryptographic Algorithms and Energy Consumption

The first step towards providing security is to have a key for encryption/decryption. Since sensor nodes are energy constrained, efficient cryptographic algorithms are necessary to maximize the battery life time of sensor nodes. The amount of computational energy consumed by a cryptographic algorithms on a given microprocessor is proportional to the number of clocks needed by the processor to compute the cryptographic algorithm [11]. Table I presents typical resources of a sensor node, which gives an insight that cryptographic primitives must be chosen carefully with respect to their code size and energy consumption, to prolong the life of a sensor node.

Given the limited resources, asymmetric cryptographic algorithms are obviously not suitable for sensor networks. Thus, symmetric key algorithms are the only other choice. However, symmetric key algorithms may

Table I. Characteristics of smart dust sensors [12].

CPU	8-bit, 4 Mhz
Storage	8 K Instruction flash 512 bytes RAM 512 bytes EEPROM
Communication	916 MHz radio
Bandwidth	10 kbps
Operating system	Tiny OS
OS code space	3500 bytes
Available code space	4500 bytes

compromise security due to small key length and memory available on the nodes. Hasan *et al.* [13] evaluated several cryptographic encryption algorithms for sensor networks. Some of the algorithms evaluated are AES Rindajel; Tiny encryption algorithm (TEA), DES and Blowfish (a mini-version of Blowfish). The results of their evaluation is summarized [13] in Table II.

Hodjat and Verbauwhede did similar studies on cryptographic algorithms [14]. The emphasis was more on ad hoc networks rather than on sensor networks. However, their experiments were performed on Wireless Integrated Network Sensor (WINS) platform developed by Rockwell Scientific. Their studies reveal that, Rijndael AES consumes 0.31–0.85 *mJ* for data encryption depending on key size. However, the energy consumption for decryption is 30% more than encryption. The difference is due to the number of shifts performed in the shift row routine and the larger GF(2^8) elements used in mix column transformation routine. On the other hand, the energy consumption for elliptic curve cryptography (ECC) were high compared to Rijndael AES. For instance, a key size of 64 for 128-bit point multiplication the energy consumptions were between 226.2–351.01 mJ.

These results give us an insight on how these cryptographic algorithms perform. However, choice of a particular algorithm is always a tradeoff between security and memory/key length. For example, Rindajel is a very algorithm but requires at least 800-byte memory space for lookup table. TEA is a small algorithm, but it is not as secure as Rindajel.

Table II. Performance analysis of cryptographic algorithms.

Algorithm	Key length (bits)	Time for 16B input (ms)	Throughput (kbps)
TEA	128	8.402185	1904.267
AES	128	7.639798	2094.296
DES	56	8.218642	1946.794
Blowfish	128	7.781995	2056.028

2.2. SPINS

Sensor Protocols for Information via Negotiation (SPINS) [15] is one of the most popular models that provides secure communication among nodes. The authors propose two security building blocks called Sensor Network Encryption Protocol (SNEP) and μTESLA. SNEP provides two-party data authentication, data confidentiality, integrity, and data freshness. μTESLA provides authentication for broadcast data.

2.2.1. SNEP

SNEP uses two counters that are shared by the two communicating parties, one for each direction. Each communicating party increments its counter after each block of data is sent. Normally, these counters are sent in the corresponding messages. The communicating parties synchronize their counter values by using a counter exchange protocol described later in this subsection.

SNEP uses a message authentication code (MAC) to provide authentication and data integrity. Consider two communicating parties A and B, which share a master key χ_{AB} with the base station. They derive independent keys from the master key using a pseudo-random function F as $K_{AB} = F_\chi(1)$, $K_{AB} = F_\chi(3)$ and MAC keys $K'_{AB} = F_\chi(2)$ and $K'_{BA} = F_\chi(4)$. Independent keys are used in both encryption and MAC operations in order to prevent introducing any weakness. The following is a representation of a message exchange from A to B.

$$A \rightarrow B : \{D\}_{<K_{AB},C_A>},$$
$$\text{MAC}\left(K'_{AB}C_A \| \{D\}_{<K_{AB},C_A>}\right) \qquad \text{(Plain SNEP)} \tag{1}$$

Here the encrypted data is $E = \{D\}_{<K_{AB},C_A>}$, where D is the data, K_{AB} is the encryption key and C is the counter; $\text{MAC}(K'_{AB}C_A \| \{D\}_{<K_{AB},C_A>})$ gives the MAC. $K'_{A,B}$ is the secret key used to generate MAC. SNEP provides data freshness, that is, it ensure that the no adversary replayed old messages. Plain SNEP provides only weak data freshness[‡] and thus can be used if only confidentiality and authentication features are required. Strong freshness[§] can be achieved using

[‡] Weak data freshness provides partial message ordering, but carries no delay information.

[§] Strong data freshness provides a total order on a request-response pair, and allows for delay estimation, it is useful for synchronization.

a nonce or a MAC with a nonce included in its computation. The complete SNEP protocol with strong freshness is as follows.

$$A \rightarrow B : N_A, R_A \tag{2}$$

$$B \rightarrow A : \{R_B\}_{<K_{BA}, C_B>}, \quad \mathrm{MAC}\left(K'_{BA}, N_A \| C_B \| \{R_B\}_{<K_{BA}, C_B>}\right) \tag{3}$$

The *counter exchange protocol* aims to synchronize the counter values shared by the communicating parties after each message is sent. Let C_A and C_B be the counter values of A and B, respectively. The counter values are initially bootstrapped using the following protocols:

$$A \rightarrow B : C_A \tag{4}$$

$$B \rightarrow A : C_B, \mathrm{MAC}\left(K'_{BA}, C_A \| C_B\right) \tag{5}$$

$$A \rightarrow B : \mathrm{MAC}\left(K'_{AB}, C_A \| C_B\right) \tag{6}$$

Note that counter values are sent in plain text. By using them as a nonce, strong freshness is achieved by the protocol under the assumption that the same counter value is not used for different runs of the protocol. Also, an additional nonce is used when a party finds the counter value of the other communicating party to be out of sync. In this case, the nonce is used to achieve strong freshness of the reply from the other party in question.

$$A \rightarrow B : C_A \tag{7}$$

$$B \rightarrow A : C_B, \mathrm{MAC}\left(K'_{BA}, C_A \| C_B\right) \tag{8}$$

SNEP offers many other features like low communication overhead, by using a common state between the two communicating parties. Also, as the counter value for a message is not the same after the message is sent, the same message can be encrypted in a different way for every message transfer. The message ordering feature achieves weak freshness and the use of a MAC provides data authentication and prevents replay attacks. By achieving semantic security, SNEP prevents eavesdroppers from interpreting the plaintext from the encrypted message.

2.2.2. μTESLA

μTESLA protocol is the *micro* version of the TESLA protocol [16] and has been developed for providing broadcast authentication on low power devices like sensor nodes. TESLA uses digital signatures for authentication which is normally too expensive for sensor nodes. Also, it has 24 bytes of overhead per packet (the key is sent in each packet), while typical sensor node messages are about 30 bytes long making it difficult to send the key with each message.

To overcome the short-comings of the TESLA protocol, the authors [15] proposed μTESLA to broadcast authenticated data across sensor nodes. Previously proposed authenticated broadcast protocols used asymmetric cryptography to prevent replay of messages from the sender. μTESLA uses symmetric keys but simulates asymmetry by disclosing the keys after a certain amount of delay. The authors have implemented μTESLA separately for two cases: one where the base station broadcasts authenticated data and another where the nodes broadcast authenticated messages.

In cases where base station is broadcasting messages, the base station uses a MAC computed on the message using a secret key. The key used is known only to the base station for the time being. Since (we assume) base station and nodes are loosely time synchronized, the nodes know an upper limit for synchronization error. Nodes also know the time slots at which keys will be disclosed. With this knowledge nodes can decide whether the key for a particular message has been disclosed or not. If the key for that message was not disclosed yet, a node can be sure that the message was not tampered in transit as only the base station has the key for the MAC of that message.

Next, nodes buffer the message until the corresponding key is revealed by the base station which broadcasts the key (for verification) to all receiving nodes. If the key is correct, the node uses it to authenticate the message stored in its buffer. If the key is incorrect or it was already revealed by the time the node received the message, the message is simply discarded as an adversary might have altered the data. Thus, loose time synchronization is a requirement between base station and nodes.

2.2.2.1. MAC key. μTESLA uses a one-way key chain; that is, each MAC key is a part of a key chain generated using a one-way public function F. The key chain is generated by choosing the first key in the chain, say K_n randomly. Next, the public function F is repeatedly applied to generate all the keys. That is, $K_i = F(K_{i-1})$. Each key in the generated key chain is associated with a time interval. Thus, the base station computes MAC of the messages in a time interval with the key of that time interval.

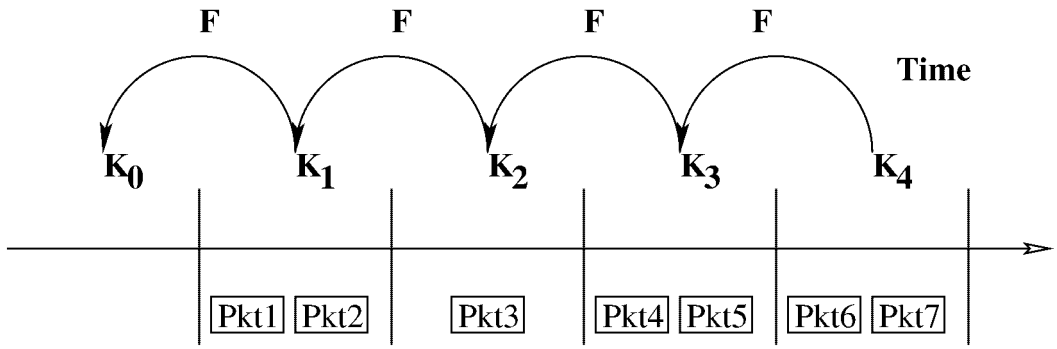

Fig. 1. Using a time-released key chain for source authentication.

Figure 1 gives an example of μTESLA's one-way key chain. Each key in the chain corresponds to a time interval; the same key is used for authenticating the packets sent within a particular interval. In this example, the key of a particular interval is disclosed after two time steps. The authors assume that the receiver knows K_0 in an authenticated manner and that it is in loose time synchronization with the sender. Packets Pkt_1 and Pkt_2 sent in interval one have a MAC with the same key K_1. Similarly, packet Pkt_3 has a MAC with K_2. As mentioned previously, the receiver cannot authenticate the packet until after two time steps have expired. Suppose the packets Pkt_4, Pkt_5, and Pkt_6 are lost, and one of these lost packets has the revealed key K_1; the receiver still cannot authenticate the received packets. However, when the base station reveals the key K_2, the receiver can authenticate Pkt_3 as well as find out the key K_1 as follows.

$$K_0 = F(F(K_2)) \tag{9}$$

$$K_1 = F(K_2) \tag{10}$$

Thus, the receiver can now authenticate all the received packets.

In the case where nodes broadcast authenticated messages, there are two mechanisms. One method is to send authenticated data using SNEP to the base station, which then re-broadcasts it to the other nodes in the network as described previously. Another method calls for a node to broadcast the message to all other nodes and the one-way key chain is stored at the base station. The base station can either send the keys in the key chain to a sender node or broadcast them to other nodes.

A detailed description of μTESLA is given in Reference [15]. An extension to μTESLA is proposed in multi-level μTESLA [17]. It is a scalable broadcast authentication scheme, which avoids unicasting the initial parameters from the base station to all nodes.

2.3. Chen *et al.*'s Model

Chen *et al.* [18] proposed a security model that provides secure communication between the base station and a node only. This model is similar to SPINS, and consist of two protocols for secure communication: one for base station to mote (a sensor node) confidentiality and authentication, and the other for the source authentication.

2.3.1. Base station to mote confidentiality and authentication protocol

Each packet includes an eight-byte MAC. The protocol uses RC5 for encryption and the MAC, as calculated based on RC5, is not easily reversible. For authentication purposes, base station uses a shared key to calculate MAC of the received packets and checks it with the MAC included in said packets. MAC can also be used to check for transmission errors, that is, it also functions as cyclic redundancy code (CRC). When a MAC check fails, the packet is sent to the (host) application with an error bit set. It is then the application's duty to decide whether to accept or reject the packet.

For confidentiality purposes, RC5 is used in OFB (output-feedback mode). An initialization vector (IV) is used along with the shared key to calculate a pad. Cipher text is obtained by XOR-ing the plaintext with the generated pad. The advantage of OFB mode is its stream cipher nature, as it generates a cipher text which is of the same size as the plain text, thus saving on the bandwidth. The entire protocol is shown below [18]. M denotes a mote and BS denotes the base station $\{\cdots\}$ denotes encryption, $(\cdots)$ denotes MAC.

$$X = \{\text{payload, seq\_no}\}_{K_{M\_\text{BS}}}$$

$$M \rightarrow \text{BS} \ : \ X(\text{src, dst, AM\_handler, } X)_{K_{M\_\text{BS}}}$$

2.3.2. Source authentication protocol

This protocol is used for mote-side (sensor node) authentication and is based on TESLA [16]. It includes a one-way key chain generated using a one-way function. Key chain generation and usage is very similar to that of μTESLA, described previously in Subsubsection 2.2.2. However, unmodified TESLA has a few drawbacks. First, the sensor node and base station have no way to share an initial TESLA key. Second, the memory within sensor nodes is insufficient to buffer packets. Therefore, TESLA is modified to fulfill its requirements to set up a secure communication. The proposed solution calls for buffered packets to be stored in sensor node on-board EEPROM. With this modification, the initial key problem can be solved by making the receiver node send a request to the base station for the initial key through a secure channel. Since the base station's reply is encrypted, receiver can trust the key and use it to bootstrap group communication.

2.4. Multi-Tiered Security Architecture

This model has been proposed by Sasha Slijepcevic *et al.* in Reference [19]. The goal of this model is to minimize security-related energy consumption by offering a range of security levels that nodes can implement. This approach is based on the principle that "data items must be protected to a degree consistent with their value" [20]. The approach relies on the classification of the types of data in sensor networks and in identifying possible communication security threats according to that classification. In this multi-tiered architecture, a different security mechanism is defined for each type of data. Each mechanism has different resource requirements and hence allows for more efficient resource management. Some of the security threats that are considered in this model are as follows.

(1) Injection of malicious code into a network can change the behavior of the network in unpredictable ways. A malicious code can destroy the whole network or can result in an adversary taking over the network.
(2) Location information of sensor nodes may help an adversary to destroy them.
(3) The level of protection of application-specific data depends on the security requirements of the particular application.
(4) Injection of false messages can give incorrect information about the environment and hence consume scarce energy resources.

2.4.1. Communication security scheme

The types of data in the network are classified as mobile code, locations of sensor nodes, and application specific data. Node use shared symmetric keys for encryption to simplify key management, but they do not offer strong authentication. Since all three types of data contain more or less confidential information, content of all messages in the network is encrypted. Security overhead and the energy consumption for each security level corresponds to the sensitivity of the encrypted information.

Three security levels are defined according to the types of data present in the network. Security level 1 is used for the most sensitive information sent through the network, for example, mobile code. Level 2 is used for location information messages, and level 3 is used for application-specific information. These can be implemented by using different algorithms for each level or by using the same algorithm with adjustable parameters. Usage of a single algorithm with adjustable parameters occupies less memory space.

All nodes in the network share an initial set of master keys. Number of keys depends on the estimated lifetime of the network. The longer the lifetime the more keys are needed in order to expose less material for "known cipher text" attacks. One of the keys from the list of master keys is considered active at any given moment. Selection of a particular key is done using an algorithm that is based on a pseudo-random number generator running at each node with the same seed.

2.4.1.1. Security level 1. The number of messages that require this level of encryption are comparatively less. For example, messages containing mobile code are less frequent than application-specific messages. Hence strong encryption is possible in spite of the resulting overhead as one can use the current master key for encryption. In order to access a network under this security level, a potential user must have a master key, a pseudo-random number generator, and a seed used by nodes.

2.4.1.2. Security level 2. This level deals with messages that contain locations of the sensor nodes, as an example. The approach is to isolate parts of the network so that a breach of security in one part does not affect the rest. As most of the messages contain somewhat sensitive information, the overhead that corresponds to encryption significantly influences the overall security overhead in the network. Since the protection level is lower for transmitted location

information than for mobile code, the probability that a key for level 2 can be broken is higher. Hence, to restrict the damage to only one part of a network, a location-based scheme is proposed.

In such a scheme the area covered by a sensor network is divided into cells. Sensor nodes within one cell share a common location-based key. Location-based key is a function of a fixed location in the cell and the current master key. The bordering region between the cells is equal to the transmission range; nodes in this region contain the keys for all adjacent cells. Thus, two nodes within transmission range of each other have a common key. It is best to divide a network area into uniform cells as it provides a fast and easy way for a node to determine its cell membership.

2.4.1.3. Security level 3. This level of encryption can be used for application-specific messages that do not need strong encryption. Also, messages of this type are generally more frequent in a network; attempting to apply a strong(er) encryption to them would deplete available energy rapidly. The key used for encryption of messages, in this security level, is derived from the current master key. A key at this level is changed whenever the master key is changed.

By considering the frequency of messages and the level of security they need, the messages can be classified and encrypted accordingly. In Reference [19], RC6 is used for encryption primarily because of an adjustable parameter (number of rounds) that directly affects its strength. Overhead of RC6 increases with the strength of encryption measured by the number of rounds. Therefore, different levels of security can be implemented by varying the number of rounds the algorithm is allowed to run.

This model assumes that messages containing location and application-specific data occur much more frequently than messages containing mobile code. Thus, by using a multi-tiered security scheme, energy consumed for encryption on all levels is the same as that consumed by encrypting all messages using the same encryption strength. Note that this model makes the assumption that sensor nodes do not leave their groups once they have been formed, and further that newly deployed nodes are not forbidden to access the messages generated before their deployment. (That is, perfect backward secrecy is not guaranteed.) In conclusion, this multi-tiered architecture provides confidentiality but does not provide strong authentication as group keys are used for communication.

3. Key Management and Distribution

Confidentiality and authentication are critical to sensor networks in order to prevent an adversary from compromising the security of the system. Due to the network's ad hoc nature, intermittent connectivity, and resource limitations, providing key management and group-level authentication is difficult. In this section we describe some mechanisms for key pre-distribution and key-management in sensor networks. Key management services must ensure that confidentiality and group-level authentication services are available to authorized parties when needed. These services should not limit the availability of a network and should not create single points of failure.

3.1. *KeEs* Protocol

KeEs protocol was proposed by Pietro *et al.*, [21]. KeEs protocol assumes that each sensor is provided with two random seeds, S_1 and S_2, each q bits long; these seeds are kept secret. Further, each sensor node is able to store an integer counter that indicates the sequence number of the current session key and has enough memory to store a limited, constant number of session keys.

The KeEs protocol is composed of two phases: a key generation phase and a key distribution/synchronization phase. In the key generation phase, a key is automatically generated by each sensor node in a time-triggered manner. The key distribution/synchronization phase provides a means for synchronization of a key among the nodes. The KeEs protocol satisfies the following security properties of a key establishment protocol.

(1) *Session key secrecy*. It should be computationally infeasible for an adversary to recover a session key. This requirement enforces implicit key authentication: only authorized users can hold the current session key.
(2) *Forward secrecy*. No *subsequent* session keys can be recovered, given that an adversary managed to recover a contiguous subset of old session keys.
(3) *Backward secrecy*. Given that an adversary managed to recover a contiguous subset of session keys, no *previous* session keys can be recovered.

Note that session key secrecy can be subjected to a *chosen plaintext* attack and therefore, to minimize the effect, periodic re-keying is performed.

3.1.1. Key generation phase

Each sensor node autonomously generates a new session key without communicating with the other sensor nodes. In order for smooth functionality of a network, all the sensor nodes are required to generate a new session key at the same time. The event to trigger key generation can be managed in a centralized or a distributed manner.

3.1.1.1. Centralized approach. In this approach a base station BS acts as a coordinator and triggers a command to generate a new session key. The decision as to when to send the command is made by BS based on some user- or operator-provided security parameters.

3.1.1.2. Distributed approach. In this approach each sensor node stores a value, μ, that drives the generation of the new session key. After μ *clock tics* have elapsed, a sensor node invokes the generation of a new session key. Due to the potentially large number of nodes in a network, tight synchronization of clocks is not possible. This protocol assumes that the *jitter* (the maximum difference among the local clocks of sensor nodes) is limited by a constant δ.

3.1.2. Key synchronization/distribution phase

Once a session key has been generated, it must be distributed. The distribution or synchronization can, as before, be done in centralized manner or in a distributed fashion.

3.1.2.1. Centralized approach. Upon receipt of a new session key generation message, each sensor node first stores the current session key and then generates a new one. This is done to ensure decryption of all messages which are assumed to be encrypted with the old key. When a node receives such a message, it will be able to decrypt it with proper session key.

3.1.2.2. Distributed approach. In this approach each sensor node generates a new session key according to its local time-out value μ. Given that all nodes might not have the same local counter value and, further, that the clock (time) difference between any two sensor nodes is bounded by δ, jitter can be calculated as $jitter = \delta/\mu$. Let Δ be the maximum time required by a message sent by some sensor node to reach some other node in the network. Jitter can be increased by a factor of Δ/μ. If each sensor node maintains $((\delta + \Delta)/\mu) + 1$ session keys, each incoming message can be decrypted.

3.2. Basic Key Pre-distribution Scheme

This scheme was proposed by Eschenauer and Gligor in [22]. A ring of keys are distributed to each sensor node. Each key ring consists of k randomly chosen keys from a large pool of P keys which is generated off-line. Due to the random choice of keys on the rings, a shared key may not exist between some pairs of nodes. If a path of nodes sharing pair wise keys exists between two nodes, say A and B, at network initialization, then A and B can use that path to exchange a key. The details of this scheme are explained next.

3.2.1. Key distribution

The *key distribution phase* consist of five off-line steps. (1) Generating a large pool of P keys and their key identifiers. (2) Select k keys randomly out of a pool of P keys in order to establish a key ring for each sensor node. (3) Loading a key ring into each node. (4) Saving key identifiers of the keys on the key ring associated with each sensor node on a trusted controller node. (5) Loading the ith controller node with the key shared by each node.

The shared-key discovery phase is carried out during network initialization in which every node discovers its neighbors, within communication range, with which it shares keys. Each sensor node broadcasts the identifiers of the keys on their rings and thus each node can check if they share a key. A *link* between two sensor nodes exists only if they share a common key; if a link exists between two nodes, all communication on that link is secured by link encryption. The *path-establishment* phase assigns a path-key to selected pairs of sensor nodes, within communication range, that do not share a key but are connected by two or more links at the end of the shared-key discovery phase.

3.2.2. Key revocation

The entire key ring of a sensor node needs to be revoked whenever it is compromised. Revocation can be done by a controller node by broadcasting a single revocation message containing a signed list of k key identifiers for the key ring to be revoked. Whenever a node receives such a message, it checks the signature of the controller node, locates the identifiers in its key ring, and removes the corresponding keys (if any). This might disrupt and break some links. These links can be reconfigured by

restarting the shared-key discovery and possibly path-key establishment phases for them.

3.2.3. Re-keying

In some cases, the lifetime of a key expires and it requires re-keying. This can be done by the self-revocation of a key by a node. After removing the expired key, the affected nodes can restart shared-key discovery and path-key establishment phases.

3.3. *q*-Composite Random Key Pre-distribution Scheme

In the basic scheme each node is required to share exactly one key with its neighbors for communication [23]. If a sensor node is captured by an adversary, all the k keys in that node's key-ring are compromised. Therefore, all the messages in a network that have been encrypted using these keys can be decrypted by the adversary. In order to avoid this and to increase the resilience of a network against a node capture, the number of keys shared by two nodes for key-setup is increased.

Let p be the probability that two nodes share a sufficient number of keys. Increasing the number of shared keys between a pair of nodes will make it exponentially harder for an intruder to break a link, even if they gain access to a complete set of key-rings. It becomes necessary to reduce the total number of keys in P so that every node shares sufficient keys with the neighboring nodes with probability of p. These two factors determine an optimal number of keys that a node has to share with other nodes.

3.3.1. Initialization and key setup

Initialization and key setup phases are quite similar to the basic scheme except that the number of keys shared by two nodes is more than one. In the initialization phase, P random keys are chosen out of the total key space. For each node, k random keys are selected from P and stored in a node's key ring. During the key-setup phase each node broadcasts a list of key identifiers for all the keys that are in its key ring. Whenever a node hears its neighbor's list of key identifiers, it checks if it shares at least q keys with the neighbor. When a node shares at least q keys with its neighbor a new communication link key K is generated as the hash of all shared keys. In this phase, key setup is not performed between nodes that share fewer than q keys.

Sharing at least be connected after the pool size is too small then the security the need for finding the optimal pool size arises.

3.3.2. Evaluation of the q-composite key distribution scheme

q-Composite key scheme offers greater resilience against node capture when the number of nodes captured is small. When a large number of nodes are compromised q-composite schemes tend to reveal larger fractions of the network to the intruder. A small scale attacks will not have any effect as the amount of additional information revealed (with such an attack) about the rest of the network is minimal. A drawback of this scheme is that, it offers no resistance against node replication[‖] because there is no limit on the number of times each key can be used and node degree is not considered. However, this scheme supports node revocation via a trusted base station similar to the approach used in the basic scheme.

3.4. Multi-Path Key Reinforcement

In multi-path key reinforcement, a link key is established through multiple paths which helps in strengthening the security of an established link. By trading off some communication overhead, this scheme can be applied in conjunction with the basic scheme (explained in previous section) to improve the resilience of the network against node capture attacks.

A node's key ring can be determined by using the initialization and key setup phase of the basic scheme. In the basic scheme, if a node is compromised an intruder will gain access to all the keys on that node's key ring. This means an intruder can decrypt all the messages that are encrypted using these keys. Therefore, compromise of one node might result in the compromise of many other nodes.

To overcome this, a link key x is updated with a random value after the initial key setup. If this update is coordinated using a direct link between two nodes, say A and B, then an adversary who was listening to the traffic between A and B, will be able to get the new communication key also. In multi-path reinforcement, key update is coordinated over multiple independent paths. The assumption made here is that enough routing information will be exchanged such that A knows all disjoint paths to B created during initial key-setup that

[‖] Replicated node are hostile nodes inserted into network after obtaining secret information (e.g., through node capture or infiltration). With this attack an adversary can populate the network with clones of captured nodes to the extent that legitimate nodes could be outnumbered and the adversary gains control of the network.

are h hops or less in length. Let j be the number of disjoint paths to B from A. A generates j random values, $v_1, v_2, \ldots, v_j$, whose lengths are the same as the length of the key. When B receives j random values, the new key can be computed by both A and B as

$$x' = x \oplus v_1 \oplus v_2 \oplus \cdots \oplus v_j \tag{11}$$

The value of the new key x' is dependent on the values of x and j random values which are sent over j paths. So, unless an adversary successfully manages to eavesdrop on all j paths, he will not know sufficient parts of the link key required to reconstruct it. The more the disjoint paths between nodes A and B the more the multi-path key reinforcement provides security for the link between A and B. However, for any given path, the probability that an adversary can eavesdrop on the path increases with the length of that path; since if one of the links on a path is insecure then the entire path is rendered insecure. It is increasingly expensive in terms of communication overhead to find multiple disjoint paths that are very long. To reduce the overhead and to increase resilience, a 2-hop multi-path key reinforcement scheme is suggested in Reference [23]. In this approach, A could exchange a neighbor list with B. Then A and B identify their common neighbors with which both of them share a key. Once they identify their common neighbors, A and B can perform reinforcement using their secure links through these common neighbors.

Multi-path key reinforcement scheme offers security at the cost of increased network overhead. This scheme gives a significant boost to network performance when implemented using the basic scheme, and offers little performance gain with q-composite scheme [23]. The cost of improved security due to multi-path key reinforcement is an added overhead in neighbor discovery and key establishment traffic. Whether this trade-off is a good one will depend on the specific application as well as the deployment density characteristics of the sensor network.

3.5. Random-Pair Wise Keys Scheme

All the schemes discussed so far do not provide any authentication; thus, a node cannot verify the identity of a node it is communicating with. For example, suppose node A shares a set of keys K with node B and they use these keys as the basis for securing a communication link. As keys can be issued multiple times out of a key pool, another node, say C, could also hold this set of secret keys K on its key ring. As A knows nothing more about B than C, it cannot ascertain if it is actually communicating with B, but not C. With this in mind, the notion of *node-to-node authentication* is defined in Reference [23] as "the ability of a node to ascertain the identity of the nodes it is communicating with."

Node-to-node authentication can be used to detect a misbehaving node, or detect node replication, for example. A misbehaving node's identity cannot be known for sure if there are no means for authentication. To detect node replication, each node must know in advance which nodes are already deployed (that are active inside the network) so that they can further reject connection attempts using that identity. By providing node-to-node authentication, sensor nodes can perform revocations on misbehaving nodes and thus can use the distributed approach instead of the centralized one for node revocation. Random-pair wise scheme has the following properties [23].

(1) *Perfect resilience against node capture*. A compromised node reveals no information about the links that it is not directly involved with.
(2) *Node-to-node identity authentication*. Nodes can verify the identities of the nodes with whom they are communicating. An adversary cannot impersonate B unless B has already been captured.
(3) *Distributed node revocation*. Misbehaving nodes can be revoked from a network by other sensor nodes without involving a base station. However, doing so incurs a small amount of communication overhead.
(4) *Resistance to node replication and generation*. The opportunity of node replication can be reduced under this scheme by introducing a small amount of memory and communication overhead.
(5) *Comparable scalability*. This scheme supports a greater number of nodes than the basic and q-composite schemes.

Node-to-node authentication can be provided if each node which holds a key x also stores the identity (ID) of the other node which also holds x. If x is used to create a secure link with another node, then both nodes are certain of the identity of one another since no other node can hold x.

3.5.1. Initialization and key-setup

Let k denotes the number of keys in each key ring, and p denotes the probability of any two nodes being able to communicate securely. In the *initialization phase*, a total of $n = k/p$ unique node identities are generated. Each node identity is matched up with k

other randomly selected distinct node IDs and a pair wise key is generated for each pair of nodes. The key is stored in both nodes' key rings, along with the ID of the other node that shares this key.

In the *key-setup* phase, each node first broadcasts its node ID to its immediate neighbors. By searching for one another's ID in their key rings, the neighboring nodes can determine if they share a common pair wise key for communication. This can be followed by a handshake mechanism to verify that the nodes do indeed have knowledge of a common key.

3.5.2. Multi-hop range extension

The effective communication range of nodes for key setup can be extended beyond physical communication range by having neighboring nodes re-broadcast the node ID for a certain number of hops. Doing so increases the number of nodes that can hear the broadcast exponentially. By increasing the effective communication radius, the number of neighbors is also increased. Therefore, the maximum supportable network size also increases. Multi-hop range extension should be used with caution as re-broadcast is performed without verification or authentication. Potential damage to a network can be reduced by limiting the number of hops of the range extension.

3.5.3. Distributed node revocation

The basic assumption of this scheme is that there exists a mechanism by which each node will be able to detect if neighbor nodes have been compromised. Whenever a node detects another misbehaving node, it broadcasts a *public vote* against the misbehaving node. If some node, say B, observes more than some threshold number t of public votes against a node A, then B breaks off all communication with A. Upon hearing these messages, base station can send messages to undeployed nodes so that they will also erase all the keys that are in A's key ring. Thus A is completely removed from the network. The set of nodes that can vote against node A are termed as A's *voting members*. The voting scheme should have the following properties [23].

(1) Compromised nodes cannot revoke arbitrary nodes.
(2) No voting member of A is able to forge another member's vote against A.
(3) Each voting member of A must be able to verify the validity of a broadcast public vote against A.
(4) Broadcast public votes from one voting member reveal no information that would allow listeners to generate additional public votes.
(5) Broadcast public votes have no replay value.
(6) Method of propagating the broadcast to cover the entire network should not be vulnerable to denial of service attack by a malicious node operating within the network.

The threshold value t is chosen low enough such that it is unlikely that any node has a degree $<t$ in the network, but high enough such that a collection of rogue nodes cannot cause the revocation of many legitimate nodes. Having every node naively re-broadcast all votes heard on the open network presents a vulnerability to denial of service attack. Hence, under the current scheme, only voting members will re-broadcast any received public votes to one another, while all other nodes ignore the broadcast message. Voting messages are transmitted in plain text, since public votes need not be secret once they are broadcast. Since there is no transmission control in an unencrypted broadcast, each voting member that first receives a correctly verified vote performs a re-broadcast of the vote a fixed number of times at varying intervals in order to maximize the probability of a successful transmission to a neighboring voting member.

Irrespective of the network size, only a fixed number of nodes have to be compromised without detection, so that a significant proportion of the network can be revoked. To prevent a widespread release of revocation keys by compromised nodes, only nodes that have established direct communication with some node B have the ability to revoke B.

Further, in order to limit the amount of node replication possible on the network, the degree of any node can be limited. The degree of the nodes can be limited to $d_{\max}$, where $d_{\max}$ is some small multiple of d, without disrupting network connectivity. The expected degree d increases slowly with graph size n. $d_{\max}$ will generally be small compared with the degree of total connectivity. A method for node-degree counting for the random-pair wise scheme may be implemented with the public-vote counting scheme. Each node contains a voting key and some way to verify other valid voting keys. Each time a given node A forms a connection with some node B, A broadcasts its voting key for B and vice-versa. Each node can thus track the degree of all k of the nodes which share pair wise keys with it and refuse to form new connections if the degree becomes too large.

4. Secure Routing

Due to energy and other resource constraints, communication between a set of sensors to a single destination should be reduced to a minimum. Aggregation techniques are often used for secure routing and have many advantages. The "data funneling method" [24] allows the network to reduce the amount of energy spent on communication setup and control—an important concern in low data-rate communication.

Instead of having an individual data stream from each sensor to a destination, there exists only one data stream from a group of sensors to that particular destination. Lossless compression of data is done using encoding information in the ordering of the sensors' packets helps in obtaining additional gains. This "coding by ordering" [24] scheme compresses data by suppressing certain readings and encoding their values in the ordering of the remaining packets.

"Routing packets along a specified curve" [26] is a new approach to forward packets in large-scale dense ad hoc networks (such as sensor networks). In this approach route paths are represented as trajectories. The trajectory that represents the route path is embedded in the packet. This allows intermediate nodes to forward the packets to those nodes that lie more or less along the trajectory. This is an efficient technique in dense networks. When standard bootstrapping or configuration services are not available, this approach serves as an effective way of implementing many network functionalities.

Designing a secure routing protocol is essential for the smooth functioning and longevity of a sensor network. Furthermore, routing must be secured in order to prevent an intruder from obstructing the propagation of correct sensor information throughout the network. There are several challenges that must be met in the design and implementation of secure routing in sensor networks. First, wireless communication among sensor nodes makes the network vulnerable to attacks such as eavesdropping, unauthorized access, spoofing, replay, and denial-of-service (DoS). Second, sensor nodes are resource poor. These resource constraints limit the degree of encryption, decryption, and authentication that can be implemented in individual sensor nodes. Third, sensor networks face the added physical security risk of being deployed in potentially hostile environments. Thus, sensor nodes may be captured and subjected to (off-line) attacks.

Some desirable properties of secure routing protocols include [27]: (1) ways to reduce the impact of misconfiguration, (2) robustness against compromised nodes and coupling with Byzantine failures, (3) ways to ensure that only legitimate nodes participate in messages forwarding, and (4) prevention of attackers from injecting bogus routes. The following sections discuss possible attacks and some mechanisms of defense, and Table III presents a summary of network layers protocols like routing along with possible attacks on them.

4.1. Network Attacks

Many sensor network routing protocols are quite simple, and for this reason they are often even more susceptible to attacks than traditional ad hoc routing protocols. Most network layer attacks against sensor networks fall into one of the following categories [25]. (a) Spoofed, altered, or replayed routing information, (b) selective forwarding, (c) sinkhole attacks, (d)

Table III. Summary of attacks against surveyed sensor network routing protocols. A bullet in a cell means the protocol is vulnerable to the corresponding attack [25].

Protocol	Relevant attacks					
	Bogus routing information	Selective forwarding	Sink holes	Wormholes	Sybil	HELLO floods
Tiny OS beaconing	•	•	•	•	•	•
Directed diffusion (and its multi-path variation)	•	•	•	•	•	•
Geographic routing (GPSR, GEAR)	•	•			•	
Minimum cost forwarding	•	•	•	•	•	•
Clustering-based (LEACH, TEEN, PEGASIS)		•				•
Rumor routing	•	•	•	•	•	
Energy conserving topology maintenance (SPAN, GAF, CEC, AFECA)	•				•	•

Sybil attacks, wormholes, HELLO flood attacks, and acknowledgment spoofing attacks.

4.1.1. Spoofed, altered, or replayed routing information

This is a straightforward attack wherein, routing information exchanged between sensor nodes is altered, spoofed, or replayed by an attacker. By doing so, attackers can redirect network traffic, change the source route by inserting extra nodes or by removing the nodes in the active path. An adversaries may also send false error messages to a sensor node and thus launch denial of service attack.

4.1.2. Selective forwarding

Malicious nodes may ensure that certain messages are not transmitted by simply forwarding a few packets and dropping the remaining one. By dropping packets, an attacker succeeds in disrupting the network operation. Such misbehavior can be hard to detect as valid nodes may, from time to time, drop packets due to congestion/collision. This attack is more powerful when adversary is included in an active path. Jamming can also cause similar effects (refer to Figure 2).

4.1.3. Sinkhole attacks

In sinkhole attack, adversary redirects all network traffic by advertising that a route through some compromised node(s) is a high-quality route. By ensuring that all traffic flows through compromised node, attacker can, using the compromised node, fabricate packets as originating from any legitimate node. This attack can also cause an adversary to forward only selected packets. Each neighboring node of the adversary forwards packets through the compromised node and also propagates them to their neighbors.

4.1.4. The Sybil attack

During a Sybil attack, an adversary presents multiple *identities* to other nodes in the network. This attack disrupts the geographic and multi-path routing protocols by causing sensor nodes to appear to be "in more than one place at once" [25]. This reduces the diversity of routes available in the network. It also diminishes the effectiveness of fault-tolerant schemes such as distributed storage, dispersity, multi-path routing, and topology maintenance.

4.1.5. Wormholes

In a wormhole attack, an attacker tunnels messages from one point in the network to another seemingly distant (or even disconnected) point. The attacker then replays those messages from the latter point. Thus, an adversary that is actually multiple hops away may appear as if it were only one or two hops away from some node. Note that this form of attack does not require any knowledge of the cryptographic keys.

Existence of a wormhole can lead to a sinkhole attack as an attacker on one "side" of the network can redirect (or control) traffic by advertising that it provides a high-quality route. A wormhole can also cause two distant nodes to believe that they are neighbors if an attacker transmits packets between them. Without some

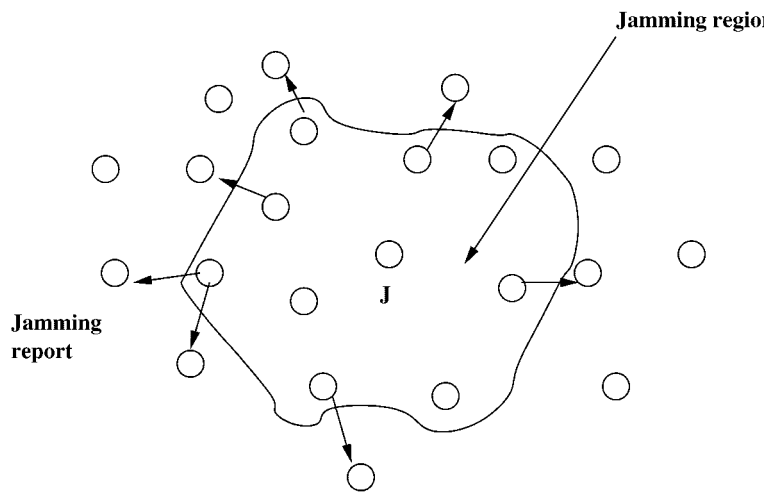

Fig. 2. Defense against a jamming attack-nodes along the edge of the jammed region report the attack to their neighbors.

mechanism to defend against wormholes, victim nodes would be unable to find routes longer than one or two hops, thus severely disrupting communication.

4.1.6. HELLO flood attack

Protocols that call for nodes to broadcast HELLO packets in order to announce their presence to neighbors are subject to this type of attack. A node that receives a compromised HELLO packet assumes that the sender and itself are within radio range of one another. The receiver then infers that the sending node must be its neighbor. However, under a HELLO attack this will not be the case. An adversary with a powerful transmitter could easily reach every node in a limited-area sensor network. Therefore, such an adversary could completely control sensor nodes' view of the network's topology.

A HELLO attack can be launched by sensor nodes within the network or by outsiders, as described above. An attacker need not construct legitimate traffic, to launch this kind of attack. An adversary can simply re-broadcast overhead packets with enough power so that each and every node in the network receives its messages. After the node realizes that a link to an adversary is false, it is left with very few options since by that time all its neighbors are possibly in communication (and "under the spell") with the adversary as well.

In some protocols, topology maintenance or flow control is based on localized information exchange between neighboring nodes. Such protocols are also prone to this type of attack.

4.1.7. Acknowledgement spoofing

In acknowledgement spoofing, an adversary spoofs link layer acknowledgment (ACK) packets of neighboring nodes. By spoofing such packets, an adversary can convince the sender that "a weak link is strong or that a dead or disabled node is alive" [25]. This encourages sender to send packets through weak links. By doing, this an adversary can launch a denial of service attack.

4.2. Counter Measure to Attacks

The attacks on sensor networks, presented previously, can be countered individually [25].

4.2.1. Outsider attacks and link layer security

Link layer security can protect a wireless network by denying access to the network itself before a user is successfully authenticated. This prevents attacks against network infrastructure, for instance. It also provides point-to-point security between directly connected network devices. Further, link layer security measures can secure frame transmissions by automating critical security operations including user authentication, frame encryption, and data integrity verification. Often, link layer security methods use a globally shared key. Many outsider-type attacks can be prevented by link layer security. It cannot, however, protect the network against wormhole attacks and HELLO flood attacks. It is also ineffective for insider-type attacks.

4.2.2. The Sybil attack

An insider participates in a network by using the identities of compromised nodes. Malicious nodes can masquerade as (often times) any other node using a globally shared key. Every node should share a unique symmetric key with a trusted base station (if one is present). Two nodes can have their identities verified and a shared key generated using a Needham–Schroeder-like protocol.

In order to construct an authenticated and encrypted link, a pair of neighboring nodes make use of the resulting key. Base stations often limit the number of neighbors a node is allowed to have. When the number of neighboring nodes crosses a pre-specified limit, the base stations broadcast an error message. This prevents an insider from wandering around a network and sharing the secret key with other nodes in the network.

4.2.3. HELLO flood attacks

Hello flood attacks can be prevented using an "identity verification protocol" [25]. This protocol verifies bi-directionality of a link between two nodes. This is done before taking meaningful action based on a message received over that link. If an adversary had a highly sensitive receiver or wormholes to multiple locations in the network, a trusted base station that limits the number of verified neighbors for each node would be able to prevent HELLO flood attacks.

4.2.4. Wormhole and sinkhole attacks

These attacks are very difficult to defend against, as wormholes make use of a private, out-of-band channel that is "invisible" to the sensor network at large. Some protocols use information such as "remaining energy or an estimate of end-to-end reliability" [25] that cannot be easily falsified to construct a routing topology. In such protocols, it is hard to prevent sinkhole attacks.

4.2.5. Selective forwarding

A compromised node has a chance of including itself in a data flow to launch a selective forwarding attack if it is located near the source; even in protocols that defend against sinkholes, wormholes, and Sybil attacks. Selective forwarding attacks are typically countered by multi-path routing schemes. It is possible to protect messages from selective forwarding attack by forwarding the messages over paths whose nodes are disjoint. However, disjoint paths are rare. In another method, an adversary can be restricted from gaining complete control of a data flow by allowing nodes to choose the next hop for packets dynamically.

4.2.6. Authenticated broadcast and flooding

Adversaries must not be able to spoof broadcast or flooded messages from any base station since these messages are considered trustworthy. Every node should be able to verify messages it receives. HELLO messages, which are broadcast by a node to its neighbors, can be authenticated and, therefore, impossible to spoof. Digital signatures can also be used for the authenticated broadcast of messages.

5. Denial of Service

"A *denial-of-service* attack is any event that diminishes or eliminates a network's capacity to perform its expected function" [28]. The goal of such an attack is to deny access to some system resource. Denial-of-service (or DoS) attacks could be passive, wherein nodes in the network force the network to cease functioning. Malicious nodes could also be responsible for DoS attacks (this is the active variety), but the intention is always the same: to prevent the network from functioning normally. DoS attacks are often a form of nuisance attacks which lead to the waste of sensor node resources. These could also be components of some other major attacks. Many of them result in improper connections and communication in a way that results in some services being rendered unavailable.

Sensor networks have limited power and battery resources so DoS attacks can potentially cause a node to cease functioning. Malicious transmissions, while actively manipulating or "poisoning data" [28], result in drained resources and thus prevent the network from getting any useful work done. Once nodes are compromised, messages can be injected into the network.

Communication protocols in sensor networks use a layered structure. Thus, from the DoS prospective, each layer is vulnerable to DoS attacks of different forms; each has different ways of handling these attacks. Table IV lists the layers of a typical sensor network and describes each layer's vulnerabilities and defenses.

5.1. Physical Layer

At the physical layer, a signal sent out by a node can be mixed with several other signals and noise, so that the actual signal is not clear. When the signal is completely jammed, DoS occurs. This consumes power and decreases network efficiency. Neighbors of nodes that have been jammed report the attack to the base station. If jamming is not sudden and complete, a node may be able to send a few "distress calls." Afterwards, the base station maps the region that is jammed and takes necessary actions.

The medium's radio signal, used for communication, is spread over a wide range of frequencies. Synchronous frequency hopping is another way to protect

Table IV. Sensor network layers, DoS attacks, and possible defenses [31].

Network layer	Attack	Defense
Physical	Jamming (refer to Figure 2)	Spread spectrum, frequency hopping, traffic detouring (Figure 3)
	Tampering	Confidentiality
Link	Collision	Integrity checks
	Resource draining	Ceiling
	Channel capturing	Small frames
Network (routing)	Abandonment of nodes	Redundancy, probing
	Location trouble	Encryption
	Misrouting	Filtering, authorization, monitoring
	Black holes	Authorization, monitoring, redundancy
Transport	Flooding	Authentication
	Synchronization trouble	Authentication

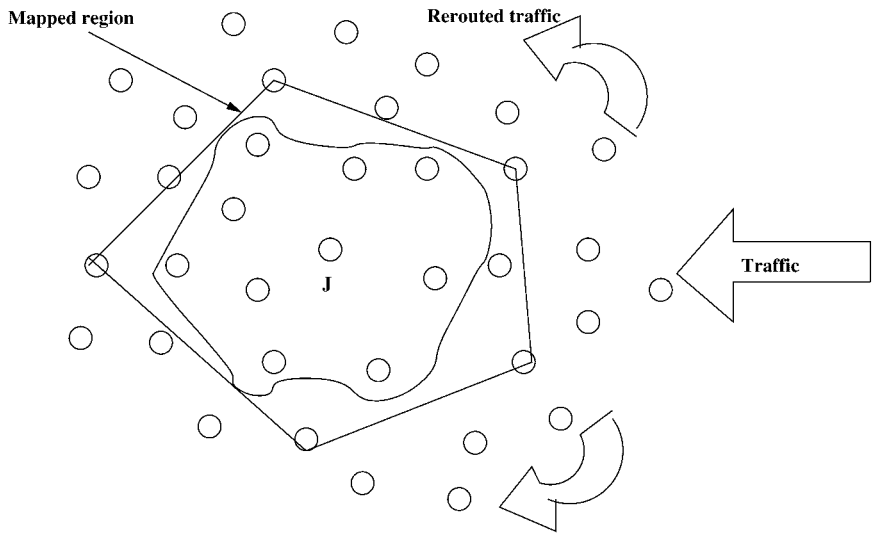

Fig. 3. Defense against jamming attack. Neighboring nodes collaborate to map the jamming reports, then re-route traffic around the jammed region.

against jamming. Spread spectrum communication avoids noise interference and transmits at a lower power density and with a narrow bandwidth. The frequency can occupy this band with no interference—reliable data can then be transmitted consistently [29].

Sensor networks can also be attacked physically. Most physical attacks involve tampering with a node physically and possibly examining or tampering with its hardware. More subtle attacks include extracting secret keys and breaking cryptosystems to gain unauthorized access to communication channels. Depending on the equipment used by an attacker, physical attacks can be of different kinds, ranging from smaller distortions to the whole network destruction.

5.2. Link Layer

Data packets are identified by sets of octets. If one octet is intruded upon, the entire packet can be disrupted. A malicious node can cause a collision in one octet if it wishes to disrupt the communication. If a collision is induced in one or more portions of the packet, then service is disturbed. Error-correcting and detecting codes, along with integrity checks, can be used to defend a network from this attack.

Sensor networks are considered to be resource-poor and need to manage energy and other resources carefully. Repeated retransmission over a short span of time would drain all the resources of the network. A limit on the number of retransmissions could ensure that the network would not be completely exhausted by these requests.

When a single node uses the entire communication channel for too long, it prevents other nodes from access to resources and starves them. This kind of unfairness can result in a DoS to most other nodes (within immediate radio range). Timeouts can be used to check that no node uses the channel for more than a fixed, limited amount of time.

5.3. Network Layer (routing)

Malicious nodes may *ignores*, *drops*, or *neglects* packets at random. This kind of a DoS attack results in communication failure between two nodes. A malicious node sends most packets and also acknowledges packets that it receives. It drops arbitrary packets. Sometimes, a node ignores messages in order to give priority to its own messages. To prevent such attacks, multiple routing paths may be used. To reduce the chances of a message never arriving at the destination, several packets of the same message can be re-transmitted.

Some sensor networks may be static for a certain period of time at a particular geographical location. In such networks, it is possible for an adversary to observe network traffic and find the location of all critical nodes and resources of the network. A passive adversary could observe this and pass on the information to an active adversary, which would then attack the network at its critical points. DoS attacks on important parts of the network and can bring the whole network down. Confidentiality of the geographical location of the network and hiding important nodes could avoid

such attacks. Further, messages that are received by an adversary can be *misdirected* or *misrouted* along some paths. If malicious nodes collude, they can tunnel routes between them, through which they pass messages along the wrong direction. A victim of this type of DoS attack is selected at random. Hierarchical routing can be used to defend against this.

Sometimes, nodes advertise a low-cost or zero-cost route within the network. This results in more traffic towards these nodes, which in turn disrupts packet delivery. This can also induce intense contention for this path, leading to bandwidth problems. Ultimately, there might be a link break in the network which *partitions* it. This kind of attack is known as a black hole attack [28,30]. Authentication of nodes and complete knowledge of the routing topology will prevent this attack.

Authorization of all nodes in the network increases defenses against DoS attacks. A trusted certification authority can be used to authenticate nodes. However, this method suffers from the usual disadvantages of centralized systems: it can prove to be a system bottleneck, and it is a single point of failure and reduces scalability. To mitigate the problem, "Watchdogs" can be used to monitor the neighbors of any node. They enforce quality of service (QoS) mechanisms that help identify misbehaving nodes. This QoS information is passed on to an authority that decides routes that are most reliable. Further details on watchdogs can be found in Reference [31].

A network can be probed to find out if nodes are forwarding messages correctly. Probes can be packets that resemble normal packets in the network. They can be interpreted in order to detect malicious nodes. They are also used to discover if messages are being transmitted or abandoned. Duplicate messages can be sent along the same path to ensure that at least one packet gets to the destination without being neglected or abandoned. If a network handles several important packets, multiple routing paths are used to guarantee that the intended message reaches the destination securely.

5.4. Transport Layer

Sensor networks have limited resources and one form of DoS is to exhaust these resources by flooding nodes with a large number of requests. An adversary can send several requests to the victim, all at once, asking for connection establishment or for other information. Resources are set aside for each request, so when the number of requests are high, the victim is drained of its resources. The node cannot (in all cases) simply reject packets that are sent to it, so even benign nodes may drop packets arbitrarily. Authentication is the first step towards preventing this problem. A ceiling must be imposed to limit the number of requests a node is permitted to receive. Doing so would prevent a complete drain of the network's resources.

In another kind of attack, an adversary can prevent two nodes from doing any useful work by introducing *synchronization* discrepancies. An adversary forges messages between two nodes, claiming lost messages and requesting retransmission. If the timing is appropriate, the nodes at the other end would get caught in an endless synchronization delay. Again, authentication of nodes is the solution to this problem.

6. Intrusion Detection

It is very important for sensors nodes and the entire sensor network to produce correct and timely results. As they are generally deployed in the "wild," sensor nodes can be "stolen" by an intruder. Once removed from network, the resource-poor node can be analyzed off-line. Such a node can later be used to launch DoS and other malicious attacks from within the sensor network. Further, breaches include attacks from outside the network, or *intrusions*, and attacks from within the network, or *misuses*. Thus, the system itself must possess ways to prevent and detect unauthorized access to system resources and data.

ID refers to the act of detecting inappropriate, incorrect, or anomalous activity within a system. An ID system gathers and analyzes information from various areas within a network to identify possible security breaches. ID systems can be classified as either *host-based* or *network-based*. Host-based ID systems operate on one or more individual hosts to detect malicious activity on the hosts. Network-based ID systems operate on the network as a whole monitoring data flows (traffic), for example.

Next we present ID system called INtrusion tolerant routing protocol for wireless Sensor NetworkS (INSENS) [32]. This system aims at tolerating intrusions rather than detecting them. An important aspect of INSENS is that though a malicious node gets control over a small number of nodes it cannot spread the damage across the entire network.

6.1. INSENS

The idea of INSENS is to design intrusion tolerant sensor networks where a single compromised node

cannot spread damage to the entire network. In order to achieve this objective, two types of attacks have to be overcome. Namely, *DoS type attacks* that flood data packets to the entire network, and *routing attacks* that can spread control packets with false routing information throughout the network. INSENS uses an asymmetric architecture consisting of a base station and sensor nodes as usually found in sensor networks. Each node shares a common secret key only with the base station. An advantage with this strategy is that if a node is compromised, an intruder will get access to only one secret key rather than the secret keys of all nodes in the network.

6.1.1. Protocol description

The INSENS protocol is built on the following three principles.

(1) Redundant approach to intrusion tolerance. This removes the need to detect compromised nodes. In other words, INSENS works satisfactorily even in the presence of undetected intruders.
(2) All computations are performed at the base station thus reducing resource consumption at other nodes.
(3) An efficient authentication mechanism is used so as to limit the extent to which damage can be done by the undetected intruders. INSENS uses symmetric key cryptography to implement these mechanisms.

The first principle points out a major problem in ID. Usually, in sensor networks, prior knowledge of the information required for anomaly-based ID is not available. Such information includes communication patterns, normal usage parameters, and so on. Obtaining this information can be quite time consuming. Also the presence of intruders makes it all the more difficult to determine these values. Besides, signature-based ID techniques cannot be used as the field is still young. Thus, implementing such techniques could expose the network to many unforeseen attacks. In view of the above reasons, the authors sought after an intrusion tolerant (as opposed to a strict prevention or detection) design strategy.

In order to avoid intruders, redundancy is included during routing. This is achieved by using multiple independent paths between each source and destination pair and by sending each message on every path to the destination. As long as there is a path that is free of intruders, one can be sure that the destination will receive an uncorrupted message. Figure 4 depicts an example of this. The second path is an intruder free (valid) path. *B* is the base station (also the destination in this case), *a* is the source node and *m* is a malicious node. The destination node determines the original uncorrupted copies by using appropriate confidentiality, integrity, and authentication techniques while exchanging messages [15].

By adhering to the second principle (from the list above), resource usage on sensor nodes is minimized. That is, key generation, building forwarding tables, and so forth, are all performed at the base station. The structure of the protocol for building forwarding table can be divided into three phases. First, a route request

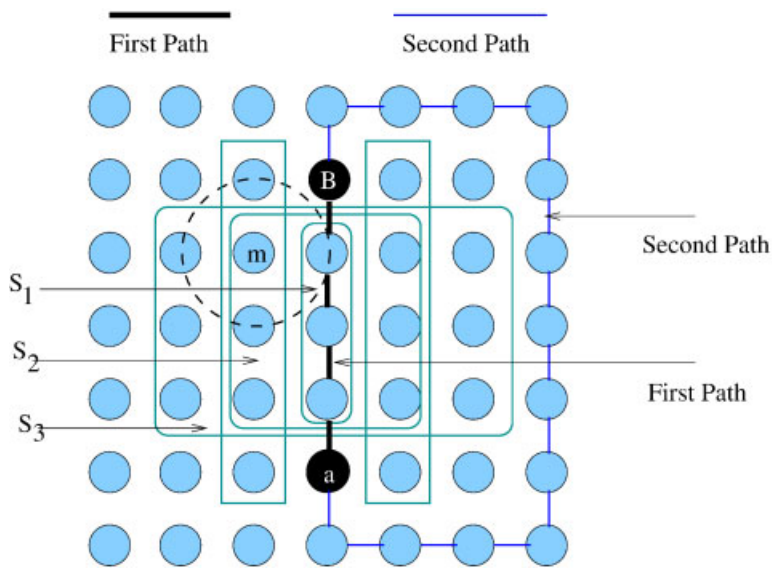

Fig. 4. Multiple routing policy.

message is sent by the base station. It then collects topology information from each sensor node. The base station finally computes and forwards the routing tables (including the redundant paths) to each node. These three phases will be dealt with in detail in the following subsections.

The authors use secure communication mechanisms such as authentication, integrity, and so on, build error-free routing tables. The authors make use of the techniques proposed by Perrig *et al.* [15]. Implementing such techniques limits the damage that can be caused due to intruders.

6.1.2. Route discovery

This phase of the protocol determines the topology of the sensor network and builds appropriate forwarding tables at each node. This process comprises three rounds: a *route request* round, a *feedback message* round, and finally a *routing table propagation* round.

6.1.2.1. Route request round. Whenever a new network is established or there is a change in the topology of the existing network due to mobility, the base station floods (in a limited sense) all the reachable sensor nodes in the network with a request message. A sensor node that receives this broadcast message for the first time simply re-broadcasts the message. Each re-broadcast message includes the path from the base station to the node that is performing the re-broadcast; that is, a request broadcast by a node a includes a path from the base station to node a along with its (the node a) own identity. Node a also adds the identity of the sender to its neighbor list. If it receives a duplicate message the identity of the sender is added to its neighbor list but the duplicate message is not re-broadcast.

By implementing the above strategy, all the nodes in the sensor network discover that the base station is collecting information about the topology of the network in order to build forwarding tables. Also, a path is established between each node and the base station which is later used in the feedback message round.

Every node has a list of its neighbors. However, a malicious node can still launch attacks in this round. It can spoof the base station by sending false messages or by including a fake path in the request message. The malicious node can also launch a DoS attack by repeatedly sending several request messages. There are two ways to overcome these attacks. One way is to use the concept of one-way (key) sequences proposed by Perrig *et al*. [15] in their μTESLA protocol. This aids in identifying the request message initiated by the base station and limits DoS-type flooding attacks. The other way is to use a keyed MAC algorithm where a MAC request at a node a is generated as

$$\mathrm{MACR}_a = \mathrm{MAC}(\mathrm{size} \mid \mathrm{path} \mid \mathrm{OWS} \mid \mathrm{type}\ , \mathrm{key}_a), \tag{12}$$

where | denotes concatenation. OWS is the one-way sequence (OWS) mentioned previously. However, a fake path in the request message can later be detected in the next round. More details on MACR can be found in Reference [32].

6.1.2.2. Route feedback round. In this round, each sensor node sends its neighbor list and the path to itself from the base station back to the base station using a feedback message. After the first round, each node waits for a certain amount of time to collect its neighbor list. The integrity of the feedback message is checked at the base station which sees if it has valid feedback messages (though incomplete due to undetected intruders).

It is possible for a malicious node to generate a tampered feedback message which passes the verification process at the base station. The inconsistency in the message can be detected by the base station after the second round but before the third round when the neighbor list of the malicious node is compared with that of the feedback message. The MACR of each neighbor is used to check for integrity.

The response to the feedback message is sent along the reverse path of the feedback request. Each child node identifies its first upstream neighbor that sent the feedback request with the current OWS as its parent. It then places the parent MACR in the feedback response packet. A casual attacker who knows just the node ID of the parent will not be able to send an erroneous feedback message because the attacker would not know a valid address of any of the upstream nodes. A clever attacker, however, would have to know the up-to-date parent MACR corresponding to the current OWS to launch an attack. Thus, MACR acts as a security function. Only if the MACRs of the feedback request and the feedback response messages (corresponding to the associated OWS) match will the upstream node know that it has been selected as a parent to the child node. One aspect of the protocol to be noticed here is that the intermediate nodes need not recalculate the MAC; they can simply do logical comparisons of MACRs. Therefore, the computation at each intermediate node is reduced.

As mentioned previously, sensor nodes, in this model, are capable of sending only unicast messages or controlled multicast messages back to the base station.

As a result, the feedback messages are subject to attacks where unicast message packets can be dropped. Such attacks are limited to the compromised node's vicinity. Also, the DoS-type attacks are avoided by not forwarding the duplicate feedback messages. This can be done based on the type and OWS field values in the path information. This, in turn, leads to two other types of attacks: *memory exhaustion attacks* and *rushing attacks* [33]. The former type of attacks can be overcome by using a single bit (such as a flag) to check for duplicate messages.

The current protocol does not defend against rushing attacks in an efficient manner. Unlike the first round where rushing must wait until an up-to-date OWS is received, an intruder need not wait for the response feedback message. An attack can be launched on the upstream nodes immediately after receiving the current OWS in the feedback request, thus causing the valid feedback response packets to be dropped as duplicates. This attack is confined only to a small number of nodes and will terminate at the base station.

Another mechanism to overcome DoS-type attacks is to use rate control to avoid flooding of feedback response messages from intermediate nodes (unlike as in the first round). Such an attack would result in congestion of the path from the compromised node to the base station as the upstream nodes do not have a method to differentiate between valid and erroneous feedback responses. By using rate control, though the malicious node floods packets at a high rate, the upstream intermediate nodes will forward them at a slower rate, thus avoiding congestion.

In order to provide confidentiality over eavesdropping, the path information and the neighbor list are encrypted using the source node's secret key. The identity of the source node is, however, left unencrypted for the base station to identify the source. As the secret key is shared only by that node and the base station, the topology information is not revealed.

6.1.2.3. Routing table propagation round. After receiving valid feedback response in the second round, the base station computes the forwarding tables for each sensor node in the network. This has the advantage that the computations are minimized on the sensor nodes, and as the base station has a complete idea of the network topology, it can decide on routes to different nodes. These routes include redundant paths that minimize the damage caused by any undetected intruders.

The route selection process is as follows. The first route is selected using Dijkstra's shortest path algorithm. The second (redundant path) is computed as follows. First, the nodes in the first path are removed from the network connectivity information (consider Figure 4 for example). The set S_1 is removed. Then the neighbors of the removed nodes, that is, set S_2, are removed. Next the neighbors of the nodes in set S_2, that is, set S_3, are also removed. Now a shortest path is computed from the updated connectivity information. If a path is found, it will be the second path else all nodes in the set S_3 are added to the network connectivity information and a shortest path is calculated. If a path is found, it will be the second path else the previous step with set S_2 is iterated. Depending on the network topology, it is also possible that a second path does not exist.

The base station computes the forwarding tables for each node once the redundant paths are computed (per node). It then propagates these tables to the respective nodes in the breadth first manner.

6.1.3. Forwarding data

Data are forwarded from source node to base station and from base station to destination node. Each entry in the forwarding table is a 3-tuple of the following format.

$\langle$*destination node, source node, intermediate sender node*$\rangle$.

For example, consider a route from S to D such as $S \rightarrow a \rightarrow b \rightarrow c \rightarrow D$. The forwarding table of node b will contain the entry $\langle D, S, a \rangle$. The intermediate node is the one that just forwarded this packet. This is included to avoid forwarding the same packet received from different intermediate nodes. For example, if h is a neighbor node of c, then h will receive a packet destined to D and forwarded by c which it should not forward. This is achieved by including the intermediate sender field in the routing table entry. Upon receiving a packet, the node searches its routing table for a matching entry. If an entry exists, it forwards the packet. Therefore, INSENS achieves intrusion tolerance by restricting the damage caused by a malicious node to its immediate vicinity.

7. Physical Security

Given that intrusion into a sensor network can involve physically attacking sensor nodes, we present a brief discussion of issues of physical security.

At a physical level there should exist some support for intrusion tolerance and support for integrity over

a secure sensor network. Intrusion tolerance [7] is concerned with physical security of the network; that is, the real elements of the network such as sensor nodes, base stations, and so on. For example, the physical insertion of malicious code (that is, via replacing a valid node with an invalid, malicious one, for instance) into a sensor node is perhaps the most dangerous physical attack. (Since sensor nodes are typically deployed redundantly within an environment, the ability of an attacker to destroy a relatively small collection of nodes is not of great concern.) Malicious code injected in the network could spread to all nodes potentially destroying the whole network or taking over the network on behalf of an adversary. A seized sensor network could send false observations about the environment to a legitimate user or send observations about the monitored area to a malicious user.

The geographical location of a sensor network should be kept as confidential as possible. This is because an intruder can easily observe the critical nodes and resources of the network and actively work towards attacking them to bring down the entire system. Denial of service attacks and intrusions at the physical level can be avoided if the locations of specific nodes are kept secret and knowledge of available resources (such as memory capacity, CPU capability, network bandwidth, and so forth) is not made public. Moreover, all resources that cannot be quickly replenished should be kept confidential. The level of (physical) security provided for should increase based on the amount of sensitive information within the network.

If the physical security of a sensor network is compromised, then the said system can *gracefully degrade* instead of completely shutting down.[¶] For example, a system could warn observers of likely occurring (or definite) trouble due to physical attacks (destroyed nodes, etc.). If a node is determined to have been physically compromised (destroyed or captured), then it should be excluded from the network to prevent data- or privacy-loss.

8. Conclusion

In this paper, we surveyed security issues, possible attacks and protocols that defend against attacks in sensor networks. We primarily focused on authentication, key management and distribution, secure routing, denial of service, and Intrusion Detection.

In terms of authentication in a sensor network, each sensor node begins "life" trusting only itself. Therefore, it is necessary for a node to be able to extend that trust to other nodes for the purpose of forming a network. This can be accomplished via various authentication protocols. Furthermore, nodes must protect themselves and the network as a whole from malicious nodes by using methods of data encryption. Pair-wise public key encryption is generally very expensive in sensor networks due to the resource constraints of sensor nodes. Therefore, research is actively focused on encryption and key management that are efficient in terms of energy and computational resources.

Any public network is prone to intruders or malicious nodes; sensor networks are no exception. Much of research has been done on secure routing in sensor networks, to defend against malicious nodes/intruders. In sensor networks, the traditional goals of secure routing, like the establishment of a path free of malicious nodes, are augmented with the need for conservation of resources. Further, sensor networks are also susceptible to denial of service attacks as well as various types of *physical* threats. Many vulnerabilities faced by sensor networks are due to the fact that sensor nodes are usually deployed "in the wild". Thus, there is a need for efficient and secure protocols at all layers of sensor protocol stack.

Acknowledgements

Authors would like to thank Venkata Deepti Kiran Bhuma, Janani Venkateswaran, and Swapna kothapally, for helping us with literature review. The paper was partially supported by the NSF Grants IIS-0242384, IIS-0324836, and CCR-0100040.

References

1. National Research Council. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*, 1st (edn). National Academy Press: Washington, DC, 2001.
2. Avancha S, Undercoffer J, Joshi A, Pinkston J. Secure sensor networks for perimeter protection. *Computer Networks* 2003; **43**(4): 421–435
3. Elson J, Estrin D. Time synchronization for wireless sensor networks. In *Proceedings of the International Parallel and Distributed Processing Symposium*, pp. 1965–1970, April 2001.
4. Akyildiz IF, Su W, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. *IEEE Communications Magazine* 2002; **40**(8): 102–114.
5. Li J, Blake C, De Couto DSJ, Lee HI, Morris R. Capacity of ad hoc wireless networks. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MOBICOM)*, pp. 61–69, July 2001.

[¶] Similar to a kernel "panic" in the UNIX operating systems.

6. Tilak S, Abu-Ghazaleh NB, Heinzelman W. A taxonomy of wireless micro-sensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review* 2002; **6**(2): 28–36.
7. Undercoffer J, Avancha S, Joshi A, Pinkston J. Security for sensor networks. Technical report, University of Maryland Baltimore County, Center for Architectures for Data-Driven Information Processing, 2002.
8. Deng J, Han R, Mishra S. Enhancing base station security in wireless sensor networks. Technical Report CU-CS 951-03, University of Colorado, Boulder, 2002.
9. Deng J, Han R, Mishra S. Securing sensor networks against base station DoS and search-and-destroy attacks. Submitted for publication (http://www.cs.colorado.edu/~mishras/research/ sensor.html), 2003.
10. Estrin D, Culler D, Pister K, Sukhatme G. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing* 2002; **1**(1): 59–69.
11. Carman DW, Kruus PS, Matt BJ. Constraints and approaches for distributed sensor network security. Technical Report 00-010, NAI Labs, 2000.
12. Pister KSJ, Kahn JM, Boser BE. Smart dust: wireless networks of millimeter-scale sensor networks. Technical report, UC Berkeley, 1999.
13. am H, Ozdemir S, Muthuavinashiappan D, Nair P. Energy-efficient security protocol for wireless sensor networks. In *IEEE VTC Fall 2003 Conference*, October 2003.
14. Verbauwhede I, Hodjat A. The energy cost of secrets in ad hoc networks. IEEE Circuits and Systems Workshop on Wireless Communications and Networking, 2002.
15. Perrig A, Szewczyk R, Wen V, Culler D, Tygar JD. SPINS: security protocols for sensor networks. *Wireless Networks* 2002; **8**(5): 521–534.
16. Perrig A, Canetti R, Tygar JD, Song D. The tesla broadcast authentication protocol. *RSA CryptoBytes* 2002; **5**(2): 2–13.
17. Liu D, Ning P. Multi-level μtesla: a broadcast authentication system for distributed sensor networks. Technical Report TR-2003-08, North Carolina State University, 2003.
18. Chen M, Cui W, Wen V, Woo A. Security and deployment issues in a sensor network. http://citeseer.nj.nec.com/chen00security.html, 2000.
19. Slijepcevic S, Potkonjak M, Tsiatsis V, Zimbeck S, Srivastava MB. On communication security in wireless ad hoc sensor networks. In *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-02)*, pp. 139–144, 2002.
20. Pfleeger CP. *Security in Computing*, 2nd edn. Prentice Hall PTR: USA, 1997.
21. Roberto Di Pietro LVM, Jajodia S. Providing secrecy in key management protocols for large wireless sensor networks. *Journal of Adhoc Networks*, To appear.
22. Eschenauer L, Gligor VD. A key-management scheme for distributed sensor networks. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)*, November 2002.
23. Chan H, Perrig A, Song D. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, 2003.
24. Petrovic D, Shah RC, Ramchandran K, Rabaey J. Data funneling: routing with aggregation and compression for wireless sensor networks. In *Proceedings of the IEEE International Workshop on Sensor Network Protocols and Applications (SNPA-03)*, May 2003.
25. Karlof C, Wagner D. Secure routing in sensor networks: Attacks and countermeasures. In *Proceedings of the IEEE International Workshop on Sensor Network Protocols and Applications (SNPA-03)*, May 2003.
26. Nath B, Niculescu D. Routing on a curve. *ACM SIGCOMM Computer Communication Review* 2003; **33**(1): 155–160.
27. Perrig A. Challenge: secure routing protocols. http://research.microsoft.com/projects/SWSecInstitute/five-minute/Perrig5.ppt, June 2003. Presentation given at Software Security: University of Washington, Microsoft Research, and Carnegie Mellon University Summer Institute.
28. Wood AD, Stankovic JA. Denial of service in sensor networks. *IEEE Computer* 2002; **35**(10): 54–62.
29. Mhoon D. License free ethernet radio modems—The only wireless ethernet modem designed for industrial environments. *Industrial Computing Magazine*, April 1999.
30. Zhou L, Haas ZJ. Securing ad hoc networks. *IEEE Network* 1999; **13**(6): 24–30.
31. Marti S, Giuli TJ, Lai K, Baker M. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pp. 255–265, 2000.
32. Deng J, Han R, Mishra S. INSENS: secure and intrusion tolerant routing for wireless sensor networks. Technical Report CU-CS 939-02, University of Colorado, Boulder, 2002.
33. Hu Y-C, Perrig A, Johnson DB. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the ACM workshop on wireless security (WiSe-03)*, pp. 30–40, 2003.

Authors' Biographies

Venkata C. Giruka received a B.tech. degree from Osmania University, Hyderabad, India in 1998. He received an M.S. degree in Computer Science from the University of Texas, Dallas, in 2001. Currently he is a Ph.D. student in the Department of Computer Science at the University of Kentucky, Lexington. His current research interests include, routing and location service for wireless ad hoc networks, security in wireless mobile networks, resource allocation in mobile distributed systems and sensor networks, and constrained optimization using evolutionary computing. He is currently a student member of the IEEE and ACM.

Mukesh Singhal is a Full Professor and Gartener Group Endowed Chair in Network Engineering in the Department of Computer Science at The University of Kentucky, Lexington. From 1986 to 2001, he was a faculty in Computer and Information Science at The Ohio State University. He received a Bachelor of Engineering degree in Electronics and Communication Engineering with high distinction from Indian Institute of Technology, Roorkee, India, in 1980 and a Ph.D. in Computer Science from University of Maryland, College Park, in May 1986. His current research interests include distributed systems, wireless and mobile computing systems, computer networks, computer security, and performance evaluation. He has published over 175 refereed articles in these areas. He has coauthored three books titled "Data and Computer Communications: Networking and Internetworking," CRC Press, 2001, "Advanced Concepts in Operating Systems," McGraw-Hill, New York, 1994 and "Readings in Distributed Computing Systems," IEEE

Computer Society Press, 1993. He is a Fellow of IEEE. He is a recipient of 2003 IEEE Technical Achievement Award. He is currently serving in the editorial board of "IEEE Trans. on Knowledge and Data Engineering" and "IEEE Trans. on Computers." From 1998 to 2001, he served as the Program Director of Operating Systems and Compilers program at National Science Foundation.

James Royalty graduated with an M.S. degree in computer science from the University of Kentucky, dexigton, USA.

Srilekha Varanasi graduated with an M.S. degree in computer science from the university of Kentucky, dexington, USA.