Secure Diffusion for Wireless Sensor Networks

Hao Yang*, Starsky H.Y. Wong†, Songwu Lu†, Lixia Zhang†

*IBM T.J. Watson Research Center
Hawthorne, NY 10532
haoyang@us.ibm.com

†UCLA Computer Science Department
Los Angeles, CA 90095
{hywong1, slu, lixia}@cs.ucla.edu

***Abstract*— Data dissemination is an indispensible protocol component for the emerging large-scale sensor networks. In this paper, we propose a secure data dissemination protocol that enhances directed diffusion to operate in the presence of compromised sensors. Our proposed solution, *Secure Diffusion*, utilizes a novel security primitive called *location-binding keys*, and exploits the available end-to-end feedback loop in Directed Diffusion. In Secure Diffusion, sensor nodes use pairwise neighbor keys to establish secure gradients, and the sink uses location-binding keys to authenticate the received sensing data. By differentiating authentic data from fabricated ones, the sink can selectively reinforce data paths and assist intermediate nodes in local reinforcement decisions to combat compromised nodes. Our security analysis shows that, in the presence of compromised nodes, Secure Diffusion can ensure both high-quality delivery of authentic data and local containment of malicious traffic.**

I. Introduction

Emerging large-scale wireless sensor networks are envisioned to be deployed in an unattended, and even adversarial, environment to support applications such as battlefield surveillance and forest fire monitoring. Because of their small size, limited processing power, and unattended deployment, individual sensor nodes are highly prone to security compromises. Therefore, it is important to build security into the network architecture and protocols, so that a sensor network can successfully operate in the presence of both component failures and malicious attacks.

In this paper, we propose novel mechanisms to secure the Directed Diffusion protocol [11]. Directed Diffusion is among the first set of data dissemination protocols developed for sensor networks; its scalability and robustness have been validated by extensive early studies [7], [8], [11], and it has been widely adopted in various deployment efforts. We believe that enhancing Directed Diffusion with security will be an important contribution to sensor networking. The design and analysis process can help us to better understand the fundamental principles in securing sensor networks. Moreover, the results can directly benefit networks that use Directed Diffusion for data delivery.

In Directed Diffusion, the data collection points, commonly called *sinks*, initiate data dissemination by flooding the query interest in the network to establish gradients at each sensor node. These gradients draw the desired sensing data down to each sink, initially at a low rate. Based on the delivery quality, the sink selects one specific path to reinforce via a hop-by-hop approach. This feedback-based data quality control is performed continuously to receive high quality of data. The application-aware sensor nodes can also perform in-network aggregation when they receive multiple flows of data.

We consider the security threats against Directed Diffusion where the adversary may potentially compromise a large number of sensor nodes. A compromised sensor may inject fabricated interests or sensing data into the network. It may also drop the interest-flooding messages or sensing data or both. Directed Diffusion can use its data quality control feedback loop to combat dropping attacks. However, the original design cannot handle false injection attacks. With such attacks in place, the user could receive incorrect data injected by the adversary; the sensor network may be abused to collect data to the benefit of the adversary; and the nodes may even be depleted of energy by large amount of fabricated interests or data injected into the network.

To eliminate the above security threats, we propose Secure Diffusion, an enhancement of Directed Diffusion that provides secure, scalable, and robust data dissemination for static and location-aware sensor networks. Secure Diffusion can ensure, in the presence of compromised nodes, both *network connectivity*, i.e., disseminating high quality of sensing data from legitimate nodes to the sink, and *containment of malicious traffic*, i.e, quarantining the traffic injected by compromised nodes within their local neighborhood.

Secure Diffusion achieves the above goals by developing a novel security primitive, called *location-binding key* (LBK), and by utilizing existing features in Directed Diffusion, i.e., the end-to-end feedback loop and the hop-by-hop neighbor relation. Secure Diffusion uses the TESLA scheme to allow for sensor nodes to authenticate interest flooding messages, and uses LBK to allow for the sink to authenticate all sensing data it receives. In addition, it also uses pairwise keys between all neighboring sensors to enable both hop-by-hop authenticated data forwarding, and maintenance of a high-quality data delivery path that adapts to bypass compromised nodes. The resulting Secure Diffusion design is a set of simple mechanisms that are readily added into the existing implementation of Direct Diffusion to secure data delivery against arbitrary numbers of compromised nodes, as proven by our analysis in Section IV.

The rest of the paper is organized as follows. In Section II we describe the sensor network model and identify our security goals in the context of Directed Diffusion. We present the Secure Diffusion design in Section III and analyze its security features in Section IV. We then discuss a few design issues in Section V, and compare Secure Diffusion to the related work in Section VI. We conclude the paper in Section VII.

II. Background

In this section, we describe our sensor network model, provide a brief overview on the Directed Diffusion paradigm, and identify our security goals.

A. Network Model

We consider a large-scale wireless sensor network in which tens of thousand of, or even more, sensor nodes are deployed in a vast terrain to monitor the remote environment. Each node is battery-powered and equipped with one or more sensors; it also has limited capabilities in terms of computation, storage, and wireless communication. An example of such sensor nodes is the current-generation MICA Mote [1] with a low-speed (4MHz) processor, small (8KB) memory, and low-rate (10Kbps) radio. The user interacts with the network through a data collection unit, called a *sink*, which can be a powerful workstation with much richer resources than ordinary sensor nodes. The user inquiry the sensor network with certain queries from the sink, and the sensor nodes whose sensing results match the query disseminate data reports back to the sink over potentially multihop wireless links.

The sensor nodes are static in that they do not move once deployed. The monitoring task typically requires each node to be aware of its geographic location to tag the sensing data. Such location-awareness can be achieved through either GPS or a localization protocol [19], [17], [24], [18]. We assume that each node can obtain its location within certain accuracy after it is deployed; the design of such a localization mechanism is out of the scope of this work.

B. Direction Diffusion

Directed Diffusion [11] is a scalable and robust data dissemination protocol developed for sensor networks. There are four components in Directed Diffusion: interest propagation, data propagation, data caching and aggregation, and reinforcement. The operations of these components can perhaps be best described by the following example.

Suppose that the user wants to know how many tanks the enemy has deployed in the geographic region *[100,100,200,200]*. Such a task is posed to the sensor nodes in the form of an interest, which is expressed in multiple attribute-value pairs, e.g., *type=tank*, and *rect=[100,100,200,200]*. The interest also has a few attribute-value pairs for control purposes, such as *duration* and *interval*. The *duration* value indicates how long the interest remains valid, while the *interval* value specifies how often the sensor node should report its sensing data.

The sink floods the network with an initial interest, which specifies a large *interval* value (e.g., *interval=1s*), to draw the sensing data slowly. Each node keeps this interest in the cache, and sets up a *gradient* of one event per second towards its neighbors from which it receives this interest. When the nodes in the specified region receive the interest, they start to send sensing data once per second. Intermediate nodes forward the data to the sink along multiple paths using the established gradients. After the sink has received the first batch of data, it decides to reinforce one neighbor along the empirically best path, e.g., the neighbor that first delivers a previously unseen event. To do so, it unicasts to the reinforced node a new interest with a smaller *interval* value, e.g., *interval=10ms*. Now this node updates its gradient as 100 events per second. Each reinforced node subsequently reinforces its own neighbors using the same localized rule. This way, one path with a much higher rate than the initial one is set up to draw higher quality of data to the sink.

Note that the gradient is defined on a per-interest basis, without any information regarding who the sink is. Thus identical interests from different sinks can be aggregated. Moreover, a sensor node may cache and aggregate the data if it receives multiple reports, and then deliver them to multiple neighbors potentially at different rates. When an reinforced path fails to deliver data at the specified rate, the intermediate node on the path that observes such degradation reinforces another neighbor for local repair.

C. Security Goals

We consider an attacker who compromises multiple sensor nodes and then mounts attacks using these nodes. Once a node is compromised, the attacker can extract all information stored on it, and would have full control over its actions. The attacker may combine the knowledge obtained from multiple compromised nodes, or could move the compromised from one location to another. However, we do not consider an attacker equipped with more powerful radios (e.g., with larger transmission range) than ordinary sensor nodes. We also assume that the attacker cannot compromise the sink or break the secure one-way function used in our design.

Directed Diffusion is vulnerable to a variety of security threats posed by such compromised nodes. Specifically, we consider the following list of attacks on Directed Diffusion:

- *Interest injection attacks*: The compromised nodes may impersonate the sink and inject unauthorized queries. As a result, the sensor network is abused to collect sensing data to answer the attacker's queries. The attacker may even deplete the energy of sensor nodes by injecting large amount of queries.
- *Data injection attacks*: The compromised nodes may inject fabricated sensing data in response to a legitimate query. Such fabricated data not only mislead the network user into skewed perception and thus wrong reactions, but waste the network resources, such as energy and bandwidth, along the delivery path.
- *Reinforcement attraction attacks*: The compromised nodes may intentionally improve their forwarding quality, e.g., by relaying the data immediately without proper backoff, in order to increase their chances to be reinforced. Once they are reinforced, they can inject fabricated data at a high rate, and suppress their neighbors in delivering legitimate data.
- *Denial of Service (DoS) attacks*: The compromised nodes may launch Denial of Service (DoS) attacks by dropping the legitimate queries or sensing data that pass through them. As a result, the network user may lose the monitoring capability and be unaware of the real events

happening in the field. The compromised nodes may also modify the legitimate queries or sensing data, which is equivalent to the combination of dropping the legitimate one and injecting a fabricated one.

- *Node impersonation attacks*: In combination of the above attacks, the compromised nodes may impersonate other legitimate nodes to avoid being detected.

To resist the above attacks, our security goal in this work is to ensure, in the presence of compromised nodes: 1) *network connectivity*, i.e., the network should deliver high-quality, authentic sensing data to reply to the user queries; 2) *local containment of malicious traffic*, i.e., the fabricated queries or sensing data injected by a compromised node should be quarantined in its local neighborhood. For simplicity, we relax the second goal in that we do not explicitly suppress a compromised node that injects fabricated data; instead, we rely on the implicit rate control and negative reinforcement (e.g., gradient timeout) mechanism in the diffusion paradigm. The downside of this choice is that compromised nodes can inject a few fabricated data in response to the initial gradients. Fortunately, the initial gradients expire rapidly, and the data rate specified by them is low. We believe that the additional complexity and overhead of explicit suppression may not justify its performance gain.

We focus on the malicious attacks against Directed Diffusion in this work. The compromised nodes may launch attacks against other components in the protocol stack, e.g., topology control, localization, and time synchronization. However, security mechanisms that protect the sensor network from such attacks are out of the scope. In particular, we assume a secure localization protocol by which sensor nodes can securely obtain their geographic locations once they are deployed.

In the proposed master-secret aware key establishment scheme, we also make an assumption that it takes longer time for an attacker to compromise a node than for the node to bootstrap itself after deployment. In other words, a sensor node is not compromised during the bootstrapping phase, in which it obtains its location securely and derives a few location-binding keys. This assumption is reasonable considering the short time for key derivation[1] and the state of the art in light-weight localization protocol design. For example, the fast localization scheme of [24] needs only five rounds of iterations. Nevertheless, we also provide an alternative key establishment scheme that is secure even though the sensor nodes may be compromised during bootstrapping. However, it incurs more communication overhead than the master-secret based scheme.

III. Design

The goal of Secure Diffusion is to provide secure, scalable and robust data dissemination, in the presence of compromised nodes, for wireless sensor networks. While Secure Diffusion inherits many design features from the original Directed Diffusion paradigm, it is crucial that the added security mechanisms should not impede the scalability and robustness properties offered by Directed Diffusion. In fact, one salient feature of Secure Diffusion is that each node keeps only localized security states (e.g., keys, secure gradients), which are independent of the global network size and robust against network dynamics. This is the key factor that contributes to the scalability and robustness of Secure Diffusion.

A. Overview

The basic building block of Secure Diffusion is a novel security primitive called *location-binding key*. Instead of associating keys with sensor nodes — the approach taken by most existing proposals, we bind symmetric keys to geographic locations independent of the underlying network topology. Specifically, we divide the terrain under scrutiny into a geographic grid, and bind multiple keys to each cell on the grid. In the bootstrapping phase, each node establishes two types of keys:

- *Sensing cell keys*: Based on its own location, the node obtains one key for each cell within its sensing range, which is also called *sensing cell*. These cell keys are used to authenticate future sensing data to the sink.
- *Neighbor keys*: The node establishes one key with each one-hop communication neighbor, and the key is bound to the locations of both nodes. The neighbors keys are used to authenticate local message (both query and sensing data) exchange.

The above localized key setup enables the basic primitives of message authentication between sensor nodes and the sink, and between neighboring nodes. On top of these primitives, Secure Diffusion combats the compromised nodes by exploiting existing features in Directed Diffusion, i.e., hop-by-hop gradients and end-to-end feedback loop of reinforcement. Specifically, Secure Diffusion enhances the Directed Diffusion with security protection against compromised nodes through the following components:

- *Secure interest propagation*: When the user interest message is flooded in the network, it is protected by a broadcast authentication scheme such as TESLA [16]. In addition, neighboring nodes authenticate to each other using their neighbor keys to establish secure gradients.
- *Secure data reporting*: The nodes whose sensing data matches the user interest disseminate data reports according to the specified rates. The nodes endorse each data report with a Message Authentication Code (MAC), which is generated using the key bound to the reported event's cell.
- *Secure reinforcement*: The sink reinforces the neighbor that delivers the best quality of authentic data, and assists the subsequent nodes in making their own reinforcement decisions by providing samples of authentic data. A round-robin scheme that mimics the Breadth-First Search (BFS) is used to reinforce a path that avoids the compromised nodes.

In the rest of this section, we will describe the design of Secure Diffusion in details. We first describe how the sensor nodes establish the location-binding keys in Section 3.2. Wen then present secure interest propagation in Section 3.3, secure

[1]Our implementation shows that it takes less than 30 ms for a node to derive one key.

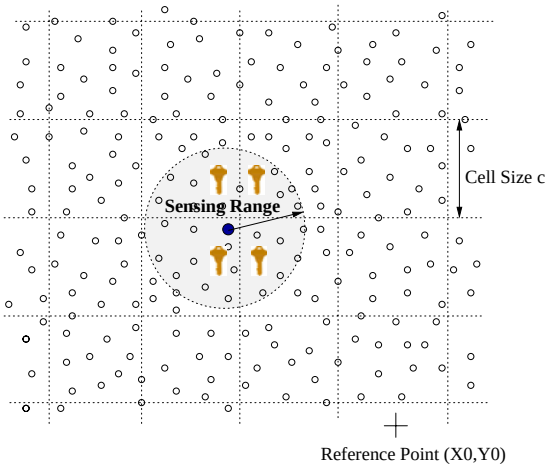

Fig. 1. Each square cell on the virtual grid is associated with multiple keys, and each node stores one key per cell in its sensing range.

data reporting in Section 3.4, and secure reinforcement in Section 3.5.

B. Location-Binding Key Establishment

To assign keys to geographic locations, we divide the terrain into a *pre-defined* grid. Note that we do not build or maintain any grid infrastructure in the network; instead, we use such a grid only to delineate cells and bind keys. The square grid can be uniquely defined by two parameters: a cell size c and an arbitrary reference point (X_0, Y_0). Accordingly, we denote a cell using the geographic location of its center (see Figure 1), which is (X_i, Y_j) such that

$$\{X_i = X_0 + i \cdot c, Y_j = Y_0 + j \cdot c; i, j = 0, \pm 1, \pm 2, \cdots\} \quad (1)$$

We bind L distinct keys to each cell on the grid. The keys of a cell are determined by its location (X_i, Y_j), together with a master secret key K_I, through a secure one-way function $H(\cdot)$[22]:

$$K_{X_i,Y_j,k} = H_{K_I}(X_i, Y_j, k), \quad (2)$$

where k is an index ranging from 1 to L.

In addition to the cell keys, there is another type of keys used in Secure Diffusion, i.e., neighbor keys. That is, two one-hop neighboring nodes share a symmetric key that is bound to the locations of both nodes. For example, if node u and node v can hear each other, the pairwise key shared between them is

$$K_{u,v} = H_{K_I}(u.ID, u.X, u.Y, v.ID, v.X, v.Y), \quad (3)$$

where $u.ID$ and $v.ID$ are the *local* identifiers[2] of the nodes, $(u.X, u.Y)$ and $(v.X, v.Y)$ denote the geographic locations of the nodes, respectively.

The sensor nodes exploit a short bootstrapping phase after deployment to obtain the cell keys and neighbor keys. We now provide two schemes for key establishment. The first scheme is very efficient yet requires each node to store the master secret for a short period of time. On the other hand, the second scheme removes the potential security threat of master secret compromise, at the cost of increased communication overhead.

1) Master-secret Aware Key Establishment: In the master-secret aware key establishment scheme, each node is preloaded with the master key K_I, as well as the grid parameters c and (X_0, Y_0). After deployment, a node obtains its own location through a localization component [24], [17]. It then determines the locations of all cells within its sensing range, which we call *sensing cell*[3], using geometry calculations. For example, given the node's sensing range R_s and its location (X, Y), at least one of the four corners in a sensing cell (X_i, Y_j) must be closer to the node than R_s, i.e., $(X_i - X \pm \frac{c}{2})^2 + (Y_j - Y \pm \frac{c}{2})^2 \leq R_s^2$. For each sensing cell, the node picks up a random seed from 1 to L, and derives *one* key for it using Equation 2. This way, each node knows exactly one key of each cell that it monitors, while nearby nodes that monitor a same cell knows different key of that cell with high chance. As we will describe later, the cell keys are used by the sensor nodes to generate MAC to endorse their sensing data.

In the bootstrapping phase, each node also establishes pairwise keys with its one-hop communication neighbors. For the purpose of local neighbor discovery, each node, say u, broadcasts a HELLO message:

$$u \rightarrow *: u.ID, u.X, u.Y, MAC_{K_I}(u.ID, u.X, u.Y)$$

$MAC_{K_I}(u.ID, u.X, u.Y)$ proves that the message is generated by one that knows K_I, i.e., a legitimate node that is not compromised. When two neighboring nodes, say u and v, are aware of each other, they can directly derive the pairwise neighbor key, defined by Equation 3, using the master secret K_I. Because both nodes know the master secret, they can reach implicit agreement on the shared key without involving any key exchange communication. The pairwise neighbor keys are used to protect future message exchange in the one-hop local neighborhood.

Finally, each node generates a unique node key, based on both its location and its local identifier:

$$K_u = H_{K_I}(u.ID, u.X, u.Y) \quad (4)$$

The node keys are used to establish neighbor keys between nodes that are deployed asynchronously, as we will describe shortly.

When the node has derived the above keys, the bootstrapping process terminates and it permanently erases the master secret K_I from its memory. That is, the nodes only store the master secret during the bootstrapping phase. Given that the keys are derived through only efficient one-way computation, the bootstrapping time is dominated by the time required by localization. In certain scenarios, e.g., with relatively dense deployment of anchor nodes[4], this period can be very short so that the attacker cannot compromise a node during the bootstrapping phase. Thus the attacker can never obtain the

[2]The local identifiers are used to facilitate the nodes to establish gradients to their neighbors. However, both Directed Diffusion and Secure Diffusion do not require global identifiers for the sensor nodes.

[3]A cell is considered as a sensing cell if any point in it resides in the node's sensing range.

[4]Most localization protocols are based on the existence of a few anchor nodes which can obtain their locations through out-of-band mechanisms such as GPS.

master secret. However, when the bootstrapping phase is not free of node compromise, the alternative scheme described in Section 3.2.3 should be used to avoid preloading the master secret onto the sensor nodes.

The master-secret aware key establishment scheme is very efficient. Each node derives and stores only a few keys: one data authentication key per sensing cell, and one pairwise key per one-hop neighbor. Moreover, all keys are derived using computationally efficient one-way functions, and no key distribution or exchange protocol is required. The only communication involved is the local broadcast of HELLO message, which is already required by most medium access control or topology control protocols for sensor networks.

2) Asynchronous Sensor Deployment: An implicit assumption in the above discussion is that the sensor nodes are deployed simultaneously, so that neighboring nodes are aware of each other before they erase the master key. However, the actual deployment of a sensor network may take a few days, and new batch of nodes may be refilled from time to time to replace those that have failed or depleted of energy. Note that each node derives the sensing cell keys and the node key independently. Therefore, the asynchronous (or incremental) sensor deployment affects only the establishment of pairwise neighbor keys.

Now we describe how two neighboring nodes that are deployed at different times can establish a pairwise key. Without any loss of generality, we consider two neighboring nodes in which one node, say node u, has already erased the master secret when the other node, say node v, is deployed. In this case, node v, in its bootstrapping phase, broadcasts a HELLO message to announce its local identifier and location. When node u receives the message, it replies with its own HELLO message to node v. Because node v knows the master secret, as well as the identifiers and locations of both nodes, it can directly derive the pairwise neighbor key shared with node u (Equation 3). Finally, node v encrypts the derived key using u's node key and sends it to node u. Now the neighbor key is established between nodes u and v. The message handshake in this process is as follows:

$$\begin{aligned} v &\rightarrow *: v.ID, v.X, v.Y \\ u &\rightarrow v: u.ID, u.X, u.Y \\ v &\rightarrow u: \{K_{u,v}\}_{K_u} \end{aligned}$$

in which $\{K_{u,v}\}_{K_u}$ denotes the encryption of $K_{u,v}$ using the node key K_u.

3) Master-secret Unaware Key Establishment: When the sensor nodes are prone to security compromise even during the bootstrapping phase, the location-binding keys should be established in a master-secret unaware manner, which trades off communication overhead for stronger security protection. In this alternative scheme, each sensor is preloaded with the grid parameters c and (X_0, Y_0), as well as a unique node key that is shared with the sink. However, the master secret K_I is not preloaded in any sensor node. Instead, the nodes establish the location-binding keys with the assistance from the sink.

Once a node obtains its geographic location, it reports the location, as well as the index of its node key, to the sink, and endorses the report with a MAC using its node key. For the routing purpose during the bootstrapping phase, the sink periodically floods a beacon message in the network to refresh a routing tree, so that each node is able to forward the location reports towards the sink. When the sink receives the location of a node, it first retrieves the node key and checks whether this key has been reported before. If the key is still fresh, it checks whether the attached MAC is correct. If the MAC is correct, the sink adds the reporting node into a map that represents where the nodes are. The sink derives the sensing cell keys on behalf of the node, encrypts these cell keys using the node key, and sends the encrypted keys back to the reporting node.

The neighbor keys can be established similarly. When a node is first added into the map, the sink checks its one-hop neighbors that are already on the map, based on the node's communication range, and then derives the neighbor keys shared between the new node and each of its known neighbors. For each neighbor key, e.g., the one shared between nodes u and v, the sink encrypts the key using the node keys of both u and v, and sends the encrypted keys to these nodes. Note that this process is asynchronous by nature, thus applies as well in cases where two nodes are deployed at different times.

The communication overhead of the above key establishment protocol can be reduced by aggregation techniques. For example, local neighboring nodes can aggregate their individual location reports, and the sink can aggregate the cell keys and neighbor keys of a node into a single message. However, due to space constraint, we will not elaborate on such optimization aspects.

4) Discussion on Location-binding Keys: The motivation of binding keys to locations comes from a critical observation: in order to detect fabricated data, one may want to know *where* the data originates, but not necessarily *who* generates the data. If the sensing data about an event "happening" at one location does not originate from that location, it must be fake: Faraway nodes simply cannot observe the events remotely due to their limited sensing range. This is exactly why location-binding keys can be used for data authentication, even though the attacker may compromise an unlimited number of sensor nodes. By using location-binding cell keys to generate MACs for the sensing data, the nodes show that they are indeed in the vicinity of the event and able to detect it. On the contrary, a compromised node does not have the keys bound to a cell outside its sensing range, thus cannot fabricate remote events at its will.

We bind keys to the locations of cells instead of individual nodes to ensure efficiency. Because the grid structure is predefined and known to all nodes, each node can independently determine the cell location, and hence the keys. In contrast, the nodes' locations have to be learned through message exchange, which incurs extra communication overhead and prolongs the critical bootstrapping phase. We bind multiple keys to each cell and use a random seed s at each node to further reduce the chance of data fabrication. Since each of nearby nodes has one, yet usually distinct, key of the cell, it can aggregate credentials to provide sufficient credibility for real events. An attacker having one or a few keys of a cell, however, cannot

fabricate enough number of credentials for non-existent events in that cell.

C. Secure Interest Propagation

The data dissemination process in Directed Diffusion starts with the flooding of an interest message in the network. As the interest is propagated, gradients are established at the sensor nodes along the paths that the interest has traversed. Therefore, security protection for interest propagation should have two mechanisms: interest authentication and gradient authentication.

In order to defend against interest injection or modification attacks, the source of an interest message should be authenticated. While digital signature can serve this purpose from the security perspective, the underlying asymmetric cryptographic primitives may not be affordable in the low-end sensor nodes; they incur heavy energy consumption in the computation. Thus we adopt the μTESLA broadcast authentication protocol [16] to authenticate the sink-to-node interest messages. The details of μTESLA protocol can be found in [16], [10]. This way, the compromised nodes cannot impersonate the sink to inject a fabricated interest or to modify a legitimate interest.

We use the neighbor keys shared between neighboring nodes to authenticate the gradients. From the data dissemination perspective, a gradient indicates both the next hop node and the forwarding data rate. While the specified data rate is carried in the interest and protected by μTESLA, we further protect the next-hop information in the gradients through local message authentication. Specifically, when a node, say node u, receives an authentic interest I, it forwards the interest to its neighbor, node v, along with a MAC generated using the neighbor key shared between them.

$$u \rightarrow v : I, u.ID, MAC_{K_{u,v}}(I, u.ID)$$

When node v receives this message, it first checks whether the interest is indeed forwarded by node u. If so, it extracts the interest and then verifies the authenticity of the interest. Otherwise, it drops the message silently. Now node v establishes a gradient to node u if and only if the interest originates from the sink and is forwarded by node u. We call such gradients that satisfy these two conditions *secure gradients*. The secure gradients not only protect the content of the user queries, but also secure the routing states (i.e., next hop and data rate) at intermediate nodes.

Eventually the interest message is flooded in the network, and all nodes establish secure gradients toward their neighbors accordingly. When the targeted sensor nodes receive the interest, they begin to disseminate their sensing data at the specified rate, as described in the next subsection.

D. Secure Data Reporting and Aggregation

In Secure Diffusion, when a sensor node reports its sensing data to the sink, it uses the cell key bound to the event's location to authenticate the data, i.e., endorsing the report with a MAC using the corresponding cell key. In the earlier example of tank detection, node s in the specified region (*rect=[100,100,200,200]*) may construct a data report D as follows:

```
type = tank // event type
location = [120,150] // event location
intensity = 0.6 // signal amplitude
timestamp = 02:10:10 // event timestamp
```

Because node s stores one key for each sensing cell, it must know one key bound to the cell in which the event is located. It uses this key, denoted by $K_{D.X,D.Y,k}$, to generate a source MAC for the data report. Node s sends the entire report (including data and source MAC) to its neighbor node t along with a second pairwise MAC, which is generated using the neighbor key shared between them:

$$\begin{aligned} s \rightarrow t : \quad & D, MAC_{K_{D.X,D.Y,k}}(D), s.ID, \\ & MAC_{K_{s,t}}(D, MAC_{K_{D.X,D.Y,k}}(D), s.ID) \end{aligned}$$

When node t receives the message, it verifies the pairwise MAC using the neighbor key. If the pairwise MAC is correct, node t then inserts the data into its local cache; otherwise, the message is silently dropped. In fact, node u sends the report to all its neighbors to which it maintains a gradient similarly.

Now the initial data starts to flow towards the sink along the gradients at intermediate nodes. When a node v receives the data from a neighboring node u, it forwards the data using its own gradients:

$$\begin{aligned} u \rightarrow v : \quad & D, MAC_{K_{D.X,D.Y,k}}(D), u.ID, \\ & MAC_{K_{u,v}}(D, MAC_{K_{D.X,D.Y,k}}(D), u.ID) \\ v \rightarrow w : \quad & D, MAC_{K_{D.X,D.Y,k}}(D), v.ID, \\ & MAC_{K_{v,w}}(D, MAC_{K_{D.X,D.Y,k}}(D), v.ID) \end{aligned}$$

An intermediate node can perform local aggregation when it sees multiple copies of the same event. For example, due to dense deployment of sensor nodes, multiple nodes can detect a tank simultaneously. While the raw data reports from these nodes may be the same, the source MACs generated by different nodes are typically different, because each node knows only one key randomly selected from L keys bound to the event's cell. Thus the intermediate node can aggregate these reports by concatenating the source MACs. Equivalently, such aggregated reports are endorsed by multiple source nodes and thus bear more credibility than the original reports with a single-source MAC.

In the Directed Diffusion paradigm, the initial interest only tasks the sensor nodes to deliver data at a very low rate. After the sink starts to receive data, it selects a few paths for reinforcement, i.e., specifying higher data rates along these paths.

E. Secure Reinforcement

One important design choice in Directed Diffusion is the localized reinforcement mechanism: Each node reinforces its neighbors using local rules only. While localized mechanism is feasible to pick up the empirically best path from the network metric (e.g., delay) perspective, it is insufficient to provide security protection because an intermediate node

cannot use local information to differentiate fabricated data from a legitimate report. Thus, when a node is reinforced, it cannot make secure decisions on which neighbors it should further reinforce. Below we describe a sink-assisted secure reinforcement scheme that preserves the localized feature of reinforcement, yet avoids the compromised nodes along the reinforced paths.

Because the sink knows the master secret K_I, it can derive all keys bound to any cell on the grid. When it receives a data report, it can check the carried source MAC to verify whether the reporting node is indeed within the vicinity of the claimed events. Obviously an incorrect MAC indicates that the report is fabricated by a compromised or misbehaving node. However, a correct MAC does not necessarily prove the authenticity of the data, because a compromised node within the interested region is able to correctly generate the MAC even though the data content is fabricated. One way to defend against such on-site compromised nodes is to exploit the redundant deployment of sensor nodes, and cross-validate the sensing data from multiple nodes. Specifically, a sink accepts a data report if and only if it is endorsed by multiple, distinct MACs.

With the capability to determine the trustworthiness of sensing data, the sink can classify its neighbors into two categories, namely *safe* and *unsafe* nodes. The safe nodes always deliver authentic data, while the unsafe nodes have delivered fabricated data at least once. If the sink has at least one safe neighbor, it limits the reinforcement choices among these safe nodes, and reinforces one of them that is empirically best, e.g., delivering the first unseen event. By induction, the neighbors that deliver data to a safe node are also safe. Therefore, each reinforced node can safely reinforce its own neighbors using the same local rules as specified in Directed Diffusion.

However, when the sink does not have any safe neighbor, the above direct reinforcement mechanism cannot be applied. Note that an unsafe node may consistently deliver fabricated data, or deliver a mix of authentic and fabricated data. Obviously a node that consistently delivers fabricated data should never be reinforced. Thus the sink reinforces one neighbor from which it receives both authentic and fabricated data. In this case, the sink picks up the best of such nodes using a networking metric, and then *selectively reinforces* it by including a sample of recently received authentic data in the reinforced interest. Each selectively reinforced node then checks its local data cache, and identifies from which neighbors it received that data sample. It keeps a list of these neighbors, ordered by the networking metrics (e.g., delay) used in Directed Diffusion. That is, the header node in the list is the empirically best one, which will be selectively reinforced. Eventually the reinforced interest reaches the targeted sensors, which then increase the rate used in disseminating their sensing data.

When selective reinforcement is used, it is possible that the compromised nodes may still be included in the reinforced path. In order to avoid the compromised nodes, each selectively reinforced node uses a round-robin strategy to maintain the list of its neighbors. Each time it receives a new selective reinforcement message, it moves the current header node to the tail, then selectively reinforces the new header node. This mimics the Breadth-First Search (BFS) algorithm, and ensures the disjointness of two consecutively reinforced paths. Therefore, as long as there exists a path between the sink and the source nodes, which is free of compromised nodes, the reinforced path will eventually converge to such a secure path.

The reinforcement mechanism is used not only for selecting a high quality path after the initial data will reach the sink, but also for handling network dynamics such as failed or degraded paths. For example, if a previously reinforced path ceases to deliver authentic data at the desired rate or starts to deliver fabricated data, the sink restarts the above reinforcement process, and the new reinforced interest is propagated to the targeted sensor nodes along a different path.

IV. Security Analysis

In this section, we analyze the security of Secure Diffusion. We first study the *network connectivity* in the presence of compromised nodes, then discuss the *containment of malicious traffic* in terms of both fabricated interests and data. Finally, we analyze the convergence speed of Secure Diffusion in finding the path that avoids compromised nodes.

A. Network Connectivity

When the adversary has compromised a large, potentially unlimited number of nodes, the first security concern is whether the network can still disseminate useful sensing data to answer the queries from the network users. Secure Diffusion can ensure network connectivity as long as the compromised nodes have not partitioned the sink from the targeted sensor nodes.

When the target sensor nodes are not isolated by the compromised nodes, they will receive the user interest which is initially flooded in the network. In such cases, there exists at least one path that connects them to the sink through multihop forwarding, yet is free of compromised nodes. As the interest is propagated, secure gradients are established along this path, and then guide the delivery of sensing data in the reverse direction to the sink. This ensures that the sink can receive authentic sensing data, despite at a low rate, during the initial phase of data dissemination. The secure reinforcement mechanism maintains the network connectivity, as shown by the following induction. The reinforced neighbor of the sink must have delivered authentic data before, and thus is connected to the targeted sensor nodes. If it is directly reinforced, all its neighbors must be delivering authentic data, and thus on at least one safe path that can reach the targeted sensor nodes. If it is selectively reinforced, its neighbors may have delivered a mix of authentic data and fabricated one. With the sample of authentic data specified in the interest, it always picks up one neighbor that has delivered authentic data at least once, i.e., having a safe path to the targeted sensor nodes. Therefore, the next reinforced node maintains the connectivity between the sink and the targeted sensor nodes.

B. Containment of Malicious Traffic

In addition to the network connectivity assurance, another important security requirement in data dissemination is to quarantine the fabricated traffic injected by the compromised nodes in their local neighborhood. Because the interest message is protected by μTESLA, the fabricated interest is never accepted or forwarded by a legitimate node. The containment of fabricated data is achieved through the reinforcement mechanism and the implicit rate control in Secure Diffusion, as discussed below.

Because a gradient specifies the maximum rate that the corresponding link should adhere to when forwarding the data, a compromised node cannot inject data faster than the gradients sent by its neighbors. In addition, each gradient is associated with a validity duration, and expires after the timeout unless explicitly reinforced. This can be considered as implicit negative reinforcement. Although the compromised nodes may inject fabricated data in the initial phase, the data rate specified by the initial gradients is very low. When the sink starts to reinforce one path, most of the compromised nodes will not be reinforced, and thus their gradients quickly expire. After that, they cannot inject fabricated data any more.

When a few compromised node injecting fabricated data are incorrectly reinforced, the entire path must be selectively reinforced. In such cases, due to the round-robin maintenance of the neighbor list at each node, two consecutively reinforced paths are completely disjoint. Thus the sink can avoid the previously reinforced nodes by restarting the reinforcement process, based on the feedback of authenticity of the data it has received. When the gradients at those incorrectly reinforced nodes expire, their fabricated data can be quarantined in the local neighborhood.

Note that with such timeout-based implicit negative reinforcement, there is a tradeoff between transient fabricated data injection and gradient refresh overhead. If a gradient is valid for a shorter period of time, the amount of data that a compromised node can inject is smaller, but the sink has to refresh the gradients at legitimate nodes more frequently. This can be improved by adaptive mechanisms for the gradient timer, which is similar to the AIMD (Additive Increase Multiplicative Decrease) in TCP congestion control. Specifically, the initial gradient timer can be set very small, in order to limit the damage caused by the compromised nodes. Before the reinforced path converges to a path free of fabricated data, the gradient timer can gradually increase to reduce the gradient refresh overhead. When a path starts to deliver authentic data only, the gradient timer can be set very large. However, when the current path starts to deliver fabricated data, we should decrease the timer and restart the reinforcement process.

C. Convergence Speed

In the previous analysis, we have shown that as long as the compromised nodes have not partitioned the targeted sensor nodes from the sink, the path reinforced by Secure Diffusion eventually converges to one that is free of compromised nodes. However, in practice, it is important to know how fast the reinforced path converges. Specifically, we evaluate the convergence speed of Secure Diffusion through the metric of *path oscillation*, i.e., the total number of paths that have been reinforced before convergence. Note that the convergence speed affects not only the communication overhead incurred by reinforcement, but also the network functionality of delivering high quality of data in a timely manner.

In general, the convergence speed depends on a number of factors, such as network topology, the number and positions of the compromised nodes, and the network metric used in local reinforcement decision. To make the analysis tractable, we consider a simplified scenario in which the nodes are randomly deployed over the terrain, and the attacker randomly compromises a fraction, denoted by α, of nodes in the network. When an intermediate node is selectively reinforced, it constructs the initial neighbor list in a random order, yet maintains the list based on the round-robin strategy described in Section III.E. Without any loss of generality, we consider two cases regarding the strategy adopted by the compromised nodes. In the first case, the compromised nodes inject fabricated data without forwarding any authentic data. In the second case, the compromised nodes not only inject fabricated data but also forward the authentic data.

When the compromised nodes do not forward authentic data, the first path reinforced by Secure Diffusion already avoids the compromised nodes, i.e., there is no path oscillation at all. This is because a sample of authentic data is attached in the selective reinforcement message. Therefore, none of the compromised nodes is reinforced.

When the compromised nodes forward both authentic and fabricated data, Secure Diffusion may not exclude them from the reinforced path in the beginning. However, due to the rate limits posed by the secure gradients, a compromised node has to drop a fraction, denoted by β, of authentic data it has received. Because a node reinforces only its neighbors that have delivered the specified sample of authentic data, the compromised nodes have less chance of being reinforced than its legitimate neighbors. Specifically, the chance that a legitimate node reinforces a compromised neighbor is:

$$P_c = \alpha(1-\beta) \tag{5}$$

Consider the end-to-end h-hop path from a legitimate source node to the sink. The chance that a reinforced path is free of compromised nodes becomes:

$$P = (1-P_c)^h = (1-\alpha(1-\beta))^h \tag{6}$$

Because each selectively reinforced node maintains its neighbor list in a round-robin manner, two consecutively reinforced paths are independent of each other. Therefore, the total number of reinforced path follows a Geometric distribution with a parameter of P. We can see that on average, Secure Diffusion reinforces

$$m = \frac{1}{P} = \frac{1}{(1-\alpha(1-\beta))^h} \tag{7}$$

paths before they converge to a path that avoids the compromised nodes.

We depict the above results in Figure 2 with a typical setting of h=20 hops. The X-axis is the fraction of compromised

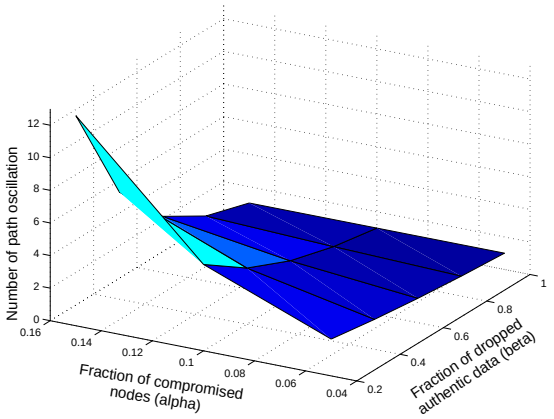

Fig. 2. Even though each node keeps only localized states, Secure Diffusion can quickly avoid the compromised nodes in the reinforced path.

nodes in the network (α), the Y-axis is the fraction of authentic data dropped by the compromised nodes (β), and the Z-axis is the corresponding number of path oscillation. We can see that even though each node keeps only localized states, Secure Diffusion can quickly avoid the compromised nodes in the reinforced path with the assistance from the sink. For example, when $\alpha = 0.1$ and $\beta = 0.5$, the *third* path reinforced in Secure Diffusion is free of compromised nodes. Not surprisingly, the number of path oscillation increases when more nodes are compromised (i.e., increasing α), or when the compromised nodes are less aggressive in injecting fabricated data (i.e., decreasing β). However, in both cases, the convergence speed of Secure Diffusion does not exhibit significant degradation, and the compromised nodes are excluded from the path after at most 12 iterations of selective reinforcement.

V. Discussion

In this section, we comment on several design issues.

Localization Security Secure Diffusion relies on the correct operation of localization protocols, so that each node obtains its location properly. However, such localization protocols also bear security vulnerabilities. In general, in order to build a truly secure sensor networking system, we believe that it is essential to secure all supporting components, such as localization and time synchronization. We plan to devise security solutions to these supporting protocols in the future.

Localization Error Location information provided by the localization protocol may contain certain degree of inaccuracies. The localization error does not affect the neighbor authentication keys or node keys; however, it may affect which cell keys a sensor node stores. Therefore, in the bootstrapping phase, the nodes should estimate their sensing regions conservatively. For instance, if the error is within $\pm e$ meters, a node can derive keys for a larger sensing range of $R_s + 2e$, in order to endorse real events.

Key Update One limitation of the current Secure Diffusion design is that it cannot update or revoke compromised location-binding keys. When new nodes are deployed to an area with compromised nodes, they may even derive some keys that are already possessed by compromised nodes. The fundamental reason is the reliance on the master secret, which is not updated. In the future we will explore methods that can allow for update of keys.

Early Data Aggregation While Secure Diffusion allows for intermediate nodes to aggregate the sensing data from different sources, it is more energy efficient to aggregate them as early as possible. In fact, the sensor nodes in the targeted region can have a local exchange protocol that uses one-hop broadcast to aggregate their sensing data into a single report, instead of delivering individual reports to the sink. However, when the communication range is less than twice of the sensing range, two nodes sensing the same event on opposite sides may not be able to communicate directly and aggregate their sensing data. In such cases, each local communication neighborhood may generate a report independently.

VI. Related Work

Key management is a fundamental challenge in a large-scale and resource-limited sensor network. A number of pair-wise symmetric key establishment schemes [3], [4], [5], [6], [13], [14], [25] have been recently proposed. Most of them use the idea of probabilistic key sharing [6] to establish trust between two nodes, each with different emphasis on enhanced security protection [3], flexibility of security requirements [25], high probability of key establishment and reduced overhead [13], or utilization of deployment knowledge [4]. Such pairwise keys can be used to authenticate a node's identity or messages; however, they cannot handle the fabricated sensing data injected by compromised nodes. Instead, semantic verification of the data is required to detect the fabricated ones. Secure Diffusion exploits location-based key management to achieve this goal. Because the data authentication keys are bound to geographic locations, the compromised nodes outside the targeted region, no matter how many there are, cannot fabricate sensing data without being detected.

Secure routing has been extensively studied in the context of ad-hoc networks [2], [10], [9], [15]. However, none of these protocols can be applied in sensor networks, because none addresses the unique feature of data-centric communication, and the network scale is limited by the excessive number of keys each node should store. The challenges of secure sensor routing are discussed in [12], together with security threat and counter-measurement analysis on a few popular routing protocols. However, it does not consider the fabricated data injection attacks launched by compromised nodes. In this paper, we focus on the Directed Diffusion paradigm and provide a complete security solution against the compromised nodes.

Two recent studies of SEF [23] and Hop-by-Hop Authentication [26] address the problem of filtering the fabricated data en-route in sensor networks. Such early drop of malicious traffic can potentially save precious energy resources at forwarding nodes. Secure Diffusion takes a different approach that quarantines the malicious traffic through implicit rate control and negative reinforcement mechanisms. As a result, Secure Diffusion is resilient to an increasing number

of compromised nodes, whereas both SEF and Hop-by-Hop Authentication completely lose security protection when the attacker has compromised beyond a small, fixed number of nodes.

There are a few recent security proposals that explicitly involve the geographic locations. The Echo protocol [20] exploits an on-site verifier node with ultrasound transceiver to verify a location claim. A recent secure routing proposal TRANS [21] monitors the behavior of static sensor nodes, and then bypasses the areas of misbehaving nodes in the route. The pairwise key establishment scheme in [14] exploits a location-aware deployment model and pre-distributes pairwise keys between nodes that are expected to be close to each other. However, Secure Diffusion differs from all these work in that it binds keys to locations, and provides a scalable secure data dissemination protocol for sensor networks.

VII. Conclusion

Securing sensor networks poses unique research challenges because of the fundamental differences between a sensor network and a traditional wireline or wireless ad-hoc network. These differences include severely resource-limited sensor nodes, very large scale, unattended deployment, and application-specific and data-centric communications. In this paper, we have presented the design of Secure Diffusion that adds critical security support to Directed Diffusion, a widely adopted data dissemination protocol for sensor networks. Secure Diffusion exploits a novel location-based approach to establishing security keys. It leverages the existing, end-to-end feedback loop in Directed Diffusion to secure the data dissemination process. In Secure Diffusion, each node keeps only a few localized keys for both neighbor authentication and node-to-sink data authentication. Based on data authenticity and quality, the sink reinforces a high-quality path, and assists the intermediate nodes to select neighbors for reinforcements using local rules. Our analysis shows that Secure Diffusion can ensure the delivery of authentic sensing data to the user, while quarantining the malicious traffic injected by the compromised nodes to their local neighborhood. Our ongoing work seeks to conduct comprehensive simulations and experimental studies to further evaluate the performance of Secure Diffusion.

References

[1] Xbow sensor networks. `http://www.xbow.com/`.

[2] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An On-Demand Secure Routing Protocol Resilent to Byzantine Failures. In *ACM WiSe*, 2002.

[3] H. Chan, A. Perrig, and D. Song. Random Key Predistribution Schemes for Sensor Networks. In *IEEE Symposium on Security and Privacy*, 2003.

[4] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge. In *IEEE INFOCOM*, 2004.

[5] W. Du, J. Deng, Y. Han, and P. Varshney. A Pairwise Key Predistribution Scheme for Wireless Sensor Networks. In *ACM CCS*, 2003.

[6] L. Eschenauer and V. Gligor. A Key-Management Scheme for Distributed Sensor Networks. In *ACM CCS*, 2002.

[7] J. Faruqu and A. Helmy. Gradient-based routing in sensor networks. *Mobile Computing and Communications Review (MC2R)*, October 2003.

[8] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly Resilient, Energy Efficient Multipath Routing in Wireless Sensor Networks. *Mobile Computing and Communications Review (MC2R)*, October 2001.

[9] Y.-C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, 2002.

[10] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks. In *ACM MOBICOM*, 2002.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM MOBICOM*, 2000.

[12] C. Karlof and D. Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. In *IEEE SPNA*, 2002.

[13] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. In *ACM CCS*, 2003.

[14] D. Liu and P. Ning. Location-Based Pairwise Key Establishments for Relatively Static Sensor Networks. In *ACM SASN*, 2003.

[15] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad Hoc Networks. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS)*, 2002.

[16] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. SPINS: Security Protocols for Sensor Networks. In *ACM MOIBCOM*, 2001.

[17] C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Annual Technical Conference*, 2002.

[18] A. Savvides, H. Park, and M. Srivastava. The n-Hop Multilateration Primitive for Node Localization Problems. *MONET*, August 2003.

[19] Y. Shang, W. Ruml, and Y. Zhang. Localization from mere connectivity. In *ACM MOBIHOC*, 2003.

[20] N. S. U. Shankar and D. Wagner. Secure verification of location claims. In *ACM WISE*, 2003.

[21] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy. Secure locations: Routing on trust and isolating compromised sensors in location-aware sensor networks. In *ACM SENSYS, Poster Abstract*, 2003.

[22] G. Tsudik. Message Authentication with One-way Hash Functions. *ACM Computer Communications Review*, 22(5):29–38, 1992.

[23] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical En-route Filtering of Injected False Data in Sensor Networks. In *IEEE INFOCOM*, 2004.

[24] F. Zhao, J. Liu, Q. Huang, and Y. Zou. Fast and Incremental Node Localization in Ad Hoc Networks. Technical Report P-2003-10265, PARC, 2003.

[25] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient Security Mechanism for Large-Scale Distributed Sensor Networks. In *ACM CCS*, 2003.

[26] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks. In *IEEE Symposium on Security and Privacy*, 2004.