On Resiliency to Compromised Nodes: A Case for Location Based Security in Sensor Networks

Hao Yang, Fan Ye, Jerry Cheng, Haiyun Luo, Songwu Lu, Lixia Zhang
Technical Report, TR040025
Computer Science Department, UCLA
E-mails: {hyang, yefan, chengje, hluo, slu, lixia}@cs.ucla.edu

Abstract

Node compromise poses severe security threats in sensor networks. Unfortunately, existing security designs can address only a small, fixed threshold of compromised nodes; the security protection completely breaks down when the threshold is exceeded. This paper studies how to achieve resiliency against an increasing number of compromised nodes. To this end, we propose a novel location-based approach where the secret keys are bound to geographic locations, and each node stores a few keys based on its own location. The location-binding property constrains the purposes for which individual keys can be used, and decouples the keys bound to different locations, thus limiting the damages caused by a group of compromised nodes. We illustrate this approach through the problem of report fabrication attacks, where compromised nodes forge non-existent events. We evaluate our approach through extensive analysis, implementation, and simulations, and demonstrate its graceful performance degradation when more and more nodes are compromised.

1 Introduction

Wireless sensor networks are ideal candidates to monitor the environment and enable a variety of applications such as military surveillance, forest fire monitoring, etc. In such a network, a large number of sensor nodes are deployed over a remote terrain to detect events of interest (e.g., enemy vehicles, forest fires), and deliver data reports over multihop paths to the user. Security is essential for these mission-critical applications working in an adverse or hostile environment.

One severe security threat in sensor networks is node compromise. Sensor nodes are typically unattended and subject to security compromise, upon which the adversary can obtain the secret keys stored in the nodes, and use the compromised nodes to launch *insider* attacks. This threat is aggravated as the adversary compromises more nodes and secret keys. Unfortunately, most existing security designs [32, 28, 4] exhibit a *threshold behavior*: The design is secure against t or less compromised nodes, but completely breaks down when

more than t nodes are compromised, where t is a fixed threshold. In reality, however, there is little constrait that prevents the attacker from compromising more than the threshold number of nodes.

In this work, our goal is to overcome the threshold limitation and achieve graceful performance degradation to *an increasing number* of compromised nodes. To this end, we exploit the static and location-aware nature of sensor nodes, and propose a novel approach of location-based security through two techniques: *location-binding keys* and *location-based key assignment*. In this approach, we bind secret keys to geographic locations, as opposed to sensor nodes, and assign such location-binding keys to nodes based on their locations. We illustrate these concepts in the context of *report fabrication attacks*, where compromised nodes forge non-existent events that cause both false alarms and network resource waste (more details in Section 2). Our design, a Location-Based Lie Detector (LBLD), demonstrates that such a location-based approach can effectively limit the damage caused by a group of compromised nodes.

In LBLD, the terrain is divided into a regular geographic grid, and each cell on the grid is associated with multiple keys. Based on its location, a node stores one key for each of its *local* neighboring cells and a few randomly chosen *remote* cells. To limit the damage of false alarms, we require that a detected event be endorsed through keys bound to the specific location of the event. An attacker compromising multiple nodes may obtain keys bound to different cells, but he cannot combine such keys to fabricate any events without being detected. To limit the damage of resource waste, each node uses its keys of remote cells to verify and drop forged reports passing through it.

Our location-based security design is highly resilient to compromised nodes for three reasons. First, it prevents the attacker from arbitrarily abusing a compromised key, because a key bound to a geographic location can only be used for purposes related to that particular location (e.g., to endorse events detected there). Second, it constraints the damage when the attacker compromises multiple nodes and accumulates their keys, because a collection of keys bound to different locations cannot be used together for any meaningful purpose. Finally, it limits the keys stored by individual nodes. because each node is randomly assigned only a few keys based on its location. As a result, the security protection offered by our design degrades gracefully, without any threshold break-down, when more and more nodes are compromised.

We have evaluated our design through extensive analysis, implementation, and simulations. The results shows that LBLD is resilient, efficient and scalable. For example, in a network of $4,000$ nodes with each node storing less than 8 keys, LBLD can drop fabricated reports after 4.2 hops on average. When the adversary have compromised 100 nodes scattered in the network, LBLD can still prevent false alarms in 99% of the field.

1.1 Related Work

Key management is a fundamental problem in large-scale, resource-limited sensor networks. Several pairwise key establishment schemes [4, 7, 8, 9, 14, 31] have

been proposed to enable mutual authentication between sensor nodes. However, these designs use node-based keys and do not handle insider attacks such as report fabrication, because the compromised nodes possess the required keying material to authenticate its messages. Our design differs from them in both the nature and the usage of secret keys.

SEF [28] and Hop-by-Hop Authentication [32] provide limited defense against report fabrication attacks through key space partitioning and interleaved authentication, respectively. However, both schemes completely lose their security protection when the number of compromised nodes exceeds a fixed threshold. In contrast, LBLD eliminates such threshold behavior and achieves graceful performance degradation against an increasing number of compromised nodes.

Numerous security solutions have been proposed to address different problems in sensor network, to name a few, secure broadcast [16], secure routing [12], secure in-network processing [17, 5], and DoS countermeasures [24]. Our design is complementary to the literature in securing the protocol stack in sensor networks.

A few recent security designs also involve geographic locations. Echo [19] uses an on-site verifier with ultrasound transceiver to verify a location claim. TRANS [22] monitors the node behavior and bypass the areas of misbehaving nodes in a routing protocol. A location-aware deployment model is used in [7] to establish pairwise keys between nearby nodes. However, to the best of our knowledge, this is the first work that exhibits graceful performance degradation and provides resiliency against an increasing number of compromised sensor nodes.

1.2 Organization

The rest of the paper is organized as follows. In Section 2 we describe our models and assumptions, and highlight the design challenges. We present our location-based security design in details in Section 3, and analyze its performance in Section 4. We further evaluate our design using implementation and simulations in Section 5, and discuss a few design issues in Section 6. We conclude the paper in Section 7.

2 Background

In this section, we describe our network, threat models and the problem context, overview an en-route filtering framework, and highlight the design challenges.

2.1 Network Model

We consider a static sensor network that monitors a remote terrain using a large number of sensor nodes. Each node is battery-powered and embedded with limited sensing, computation and wireless communication capabilities. The user queries the network through a data collection unit, called the *sink*, which can

be a powerful workstation. The nodes detect events of interest in the field, and deliver data reports over multihop paths to the sink. The sensor deployment is dense to support fine-grained collaborative sensing and provide robustness against node failures.

We assume that a sensor node can obtain its geographic location soon after it is deployed. Sensor localization is required by many applications to determine the locations of the events, and is currently an active research field [20, 21, 30]. We also assume that some form of geographic routing [10, 13] is used to deliver the reports to the sink. There are only a small number of static sinks in the network, and their locations are known when the nodes are deployed. Finally, we assume that a rough estimation on the size and shape of the terrain is known a prior.

2.2 Threat Model

We consider an attacker who can compromise multiple nodes in the network, and we do not impose any bound on the number of compromised nodes. The attacker can extract all secret keys, data, and codes stored on a compromised node, and have full control over its actions. The attacker can aggregate the secret keys obtained from multiple nodes, and store these keys back into each node. We term this as *collusion* among compromised nodes.

We assume that the attacker cannot compromise the sink, and it takes him longer time to compromise a node than it takes for a node to bootstrap itself, including obtaining its location and deriving location-binding keys. Given the state-of-art fast localization protocols [10], this can be done in a short period of time because key derivation only requires local computation[1]. Similar assumptions are also adopted in an earlier proposal [31]. We also assume that the localization protocol is secure.

As a show-case example, we focus on one specific attack, namely *report fabrication attack*, in which the compromised nodes inject many fabricated reports on non-existent events into the network. Such forged reports not only cause false alarms that deceit the application into wrong reactions , but also waste network resources (e.g., energy) along the forwarding paths. Note that the forged events could "appear" not only where nodes are compromised, but also at *arbitrary* locations because the compromised node can pretend to be "forwarding" the reports. We do not consider other attacks such as DoS [24] (e.g., channel jamming, packet dropping) in this work. More security analysis is provided in Section 4.

2.3 En-route Filtering

We adopt a general en-route filtering framework to defend against event fabrication attacks. The injected bogus reports are dropped en-route as they traverse the network to save network resources, and the eluded ones are rejected at the sink to prevent false alarms. In this framework, each node is assigned several

[1]Our implementation shows that it takes less than 3 seconds for a node to derive all keys.

symmetric keys. The node uses its keys to endorse its own reports and verify other nodes' reports passing through itself. The sink further verifies any reports that it receives. The en-route filtering part of our design, as well as SEF [28] and Hop-by-Hop Authentication [32], are all specific instances using different approaches under this framework.

2.4 Challenges

In the above framework, there exists a fundamental tradeoff between the en-route filtering power and the resiliency to compromised nodes. Intuitively, to increase the filtering power, each node should store more keys so that it has larger chances to detect and drop fabricated reports. However, to improve the resiliency, each node should store less keys to minimize the damage of compromise nodes, which may abuse their keys to endorse bogus reports. How to resolve this conflict and achieve both resiliency and effective en-route filtering is the primary challenge that our design faces.

The characteristics of sensor networks also pose scalability and efficiency challenges on our design. The design should scale well to a large network (e.g., with tens of thousands of sensor nodes), and work well with low-end sensor nodes that have stringent computation and storage limitations.

3 Design

In this section, we present the design of a Location-Based Lie Detector (LBLD) that defends against report fabrication attacks. LBLD achieves resiliency to compromised nodes through two novel techniques: *location-binding keys* and *location-based key assignment*.

As shown in Figure 1, we bind multiple keys to each cell on a geographic grid that covers the terrain, and assign these location-binding keys to nodes in a location-based manner. Specifically, each node stores *one* key for each local cell in its sensing range, which is used to endorse its own reports. In addition, each node also stores *one* key for each of a few randomly chosen remote cells, which is used to verify the reports passing through it. A legitimate report is endorsed by multiple distinct MACs, jointly generated by the detecting nodes using the keys bound to the event's cell. The forwarding nodes verify the MACs based on the claimed event location, and probabilistically filter the fabricated reports.

The rest of this section will answer the following questions: How are keys bound to locations, and what are their advantages? How are the keys assigned to nodes in a location-based manner? How are the reports generated? How are the fabricated reports filtered?

3.1 Location-Binding Keys

In order to assign keys, we divide the terrain into a *virtual*, *pre-defined* geographic square grid. Note that we do not build or maintain any grid infrastruc-

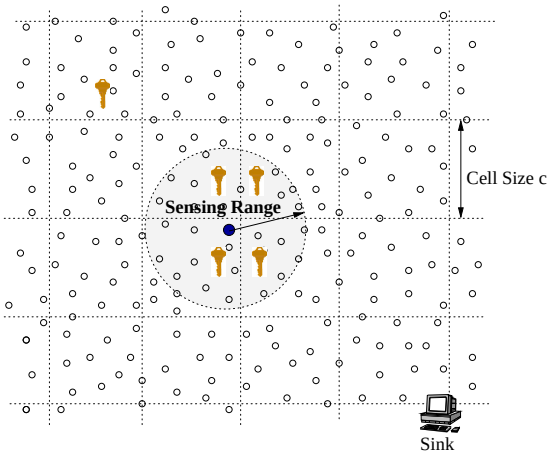

Figure 1: Each square cell on the geographic grid is associated with multiple keys. Each node stores a few local and remote cell keys based on its own location.

ture [27] in the network; instead, we only use the grid to delineate cells and bind keys. The square grid is uniquely defined by two parameters: a cell size c and a reference point (X_0, Y_0). Accordingly, we denote a cell by the location of its center (see Figure 1), which is (X_i, Y_j) such that

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

We bind L distinct keys to each cell on the grid. The keys of a cell are determined by the cell location (X_i, Y_j), together with L master secret keys K_s^I, through a secure one-way function $H(\cdot)$[23]:

$$K_{X_i,Y_j,s} = H_{K_s^I}(X_i||Y_j) \tag{1}$$

where s is an index ranging from 1 to L, and $||$ denotes concatenation.

3.1.1 Why Location-binding Keys

The motivation of location-binding keys is to prevent the compromised nodes from abusing their keys. As described later, each report must be endorsed by multiple distinct MACs using keys bound to the claimed event location. To successfully inject a bogus report, the attacker must collect enough keys from a single cell, because a collection of keys from different cells are useless. This is extremely difficult due to the randomized key assignment scheme (Section 3.2). More importantly, even when the attacker has collected enough keys from one cell, he can only fabricate reports "appearing" in that particular cell. Such constraints not only reveal diagnostic information to the sink, but also quarantine the damaged area, in terms of false alarms, in the field. This is exactly why our design can be highly resilient to node compromise.

In contrast, the traditional node-based keys are not as effective because there is no such constraints on the key usage and accumulation. When a report is required to be endorsed by multiple, say t, nodes, a group of $t+1$ or more compromised nodes can freely collude to inject any reports on bogus events "happening" at arbitrary locations. One may think that a complete map of each node's location could remedy this problem. However, in a large-scale sensor network, the nodes may not afford the storage overhead of such a map, and the construction and maintenance of this map are also very expensive.

We bind keys to cells instead of the locations of individual nodes, so that each node can independently derive, without any message exchange, the keys based on the pre-defined grid structure. The cell size c is an important parameter, which affects both the key storage overhead and the impact of compromised nodes. We will analyze this in Section 4.

3.1.2 Deriving Keys from Locations

We exploit a short bootstrapping phase to allow a node to derive keys from locations. Before deployment, each node is preloaded with one of the L master secrets, K_s^I, as well as the grid parameters c and (X_0, Y_0). Once it is deployed (e.g., via aerial scattering), it first obtains its geographic location through a localization component [30, 20]. It then uses the location-based key assignment scheme, which we will describe shortly, to pick up a few cells from the grid, and derives *one* key for each cell using Equation 1. This ends the bootstrapping phase, and the node permanently removes the master secret from its memory [31].

The key derivation is efficient because it involves only local computation of light-weight one-way functions, without any message exchange. This makes the bootstrapping phase very fast, so that the attacker cannot compromise a node before it finishes the bootstrapping. Therefore, the attacker never knows the master secrets, and as long as the program loaded in sensor nodes works correctly, a node will not derive additional cell keys.

3.2 Location-based Key Assignment

The cell keys that a node derives and stores in the bootstrapping phase are chosen based on its own location. The node stores two types of keys. One type is for the local cells within its sensing range, called *sensing cells*[2]. The node determines its sensing cells based on its sensing range R_s, cell size c and its location, and derives one key for each sensing cell. These keys will be used to endorse the events the node itself observes. The node also randomly picks up a few remote cells, called *verifiable cells*, and derives one key for each of them. These keys will be used by the node to verify the reports passing through it.

Now we describe which strategy the node should use in selecting its verifiable cells. In the absence of any *a priori* information, perhaps the best strategy is

[2] A cell is a sensing cell if there exists a point in the cell that is covered in the node's sensing range.

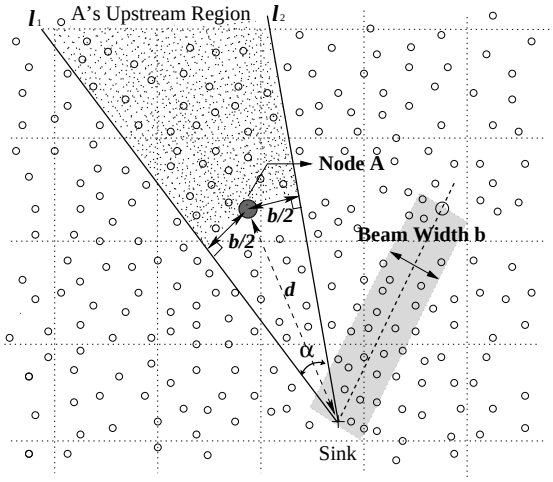

Figure 2: A report is forwarded inside a beam of width b from the source to the sink. Thus each node can estimate its upstream region based on the locations of itself and the sink.

to uniformly randomly select them from the grid. However, when the sink is *static* and geographic routing protocols [10, 13] are used to deliver packets, we can adopt more intelligent strategy to allow each node to store less keys, while retaining the filtering power. Note that such reduction has two-fold benefits. It decreases the key storage overhead yet retains the desirable filtering power, and more importantly, it improves the system resiliency by limiting the keys exposed to individual nodes.

The idea is that if a node can estimate its upstream region, i.e., those remote cells for which it may potentially forward reports, it only needs to pick up a few verifiable cells from this region. Below we describe how this can be achieved.

3.2.1 Estimating Upstream Region

We exploit the properties of geographic routing to estimate the upstream region of a node. Such routing protocols typically forward packets in a greedy manner whenever possible, and use perimeter routing to bypass the "holes" encountered. The resulted forwarding paths statistically center around a straight line connecting the source to the sink. Therefore, we model the forwarding path using a general beam model. In this model, the possible forward paths form a beam, the width of which is b, connecting from the source to the sink (illustrated in Figure 2). The intuition here is to accommodate a "hole" up to a diameter of b in the forwarding path. We will validate this model and discuss its impact in Section 5.

With the beam model, a node can estimate its upstream region using simple geometry. As illustrated in Figure 2, a node A's upstream region is the shaded

area between the two radiating lines l_1 and l_2, and away from the sink. From geometry we can calculate the spanning angle between l_1 and l_2 as

$$\alpha = 2 \arcsin \frac{b}{max(b, 2d)} \tag{2}$$

where d is the distance from node A to the sink. We can see that a node's upstream region depends on the locations of both itself and the sink. In general, a node closer to the sink has a larger upstream region. This is simply because reports originating from different places will converge around the sink.

3.2.2 Random Selection of Verifiable Cells

After a node determines its upstream region, it randomly selects a few cells from this region as verifiable cells. Specifically, a node picks up a cell as its verifiable cell with a probability of

$$P = \frac{d}{D_{max}} \tag{3}$$

where d is the node's distance to the sink, and D_{max} is the maximum distance between network edge and the sink. Given that the terrain size and shape are known, D_{max} can be preloaded in the nodes.

There are several features in the above random distribution that are worthy mentioning. First, the verifiable cells are chosen in a probabilistic manner. This is because any deterministic scheme could be exploited by the attacker: If he knows which verifiable cell a node selects, he can fool the node by fabricating events in other locations. Second, the probability is decided solely by the node's location. That is, all cells in the upstream region are considered equally important and chosen with the same probability. This maximizes the *worst-case* filtering performance. For any non-uniform strategy that gives more chance to some upstream cells and less to others, the attacker can increase his chance of success by fabricating events in the unfavored cells. Finally, the probability increases as the node becomes farther away to the sink. This is because such a node has a smaller upstream region, so it can pick up verifiable cells at a higher probability without exceeding its storage budget.

We point out that Equation 3 is only one possible strategy in selecting verifiable cells; a full exploration is yet to be carried out. However, our evaluation results show that it can indeed provide effective filtering, strong resilience, with only moderate overhead.

3.3 Report Generation

To be forwarded and accepted, a legitimate report must carry m distinct MACs, generated using keys bound to the event's cell. Because every node has one key for each of its sensing cells, it can endorse any events itself has observed. When a real event occurs, multiple detecting nodes can jointly generate the required m MACs.

The detecting nodes first reach agreement on the event description, including the event's location, through techniques such as [26, 29]. After that, each node independently generates a MAC using its own key bound to the event's cell, and broadcasts a tuple $\{s, MAC_s\}$, where s is the key index (between 1 and L). Each node also records all such tuples announced by its neighbors, and constructs a complete report after it has received m distinct MAC tuples.

Nodes overhear to avoid duplicate reports. Each node sets a random timer, and the first node that fires the timer sends out the final report to the sink. An overhearing node checks the report and updates a counter about how many tuples in its overheard list are sent. If the counter reaches m, it cancels the timer because there are enough MACs in the report. Otherwise, upon timer expiration, it sends out its own report that carries m distinct MAC tuples, with higher priority on the unused ones.

Note that a legitimate node participates in report generation only when it has sensed the event by itself. Thus a compromised node cannot deceit its neighbors into endorsing a fabricated report. The number of MACs in a report, m, provides a tradeoff between overhead and security strength, which will be evaluated in Section 4.

3.4 En-route and Sink Filtering

When an intermediate node receives a report, it verifies the report as follows: It first checks whether the report carries m distinct MACs and indexes. It then retrieves the event's location from the report, and checks whether the event resides in its upstream region. If the event's cell happens to be its verifiable cell, i.e., it has a key for that cell, it checks the carried MACs when its own key index appears in the report. It drops the report when any of the above checks fail. Otherwise, it forwards the report as usual.

The probabilistic en-route filtering cannot guarantee to drop all fabricated reports. The sink serves as the final guard in verifying and rejecting any eluded ones. Because the sink knows all master secrets, it can derive any cell key. When it receives a report, it retrieves the event's location, derives the cell keys, and checks how many correct MACs are carried. Note that there might be multiple reports for a same event. The sink decides whether to accept the event based on the total number of correct MACs it has received. If this number reaches m, the event is accepted; otherwise it is rejected.

4 Analysis

In this section we analyze the performance of our design. We start with the filtering power of LBLD against single compromised node, then analyze its resiliency when more and more nodes are compromised. We also provide an overhead analysis and a security analysis on relevant attacks, and discuss on the impact of several design para-meters. Our analysis results quantify the resiliency, efficiency, and scalability of LBLD.

	Meaning	Default
N	total number of nodes in the network	4K $\sim$ 400K
R	radius of the circular terrain	1Km $\sim$ 10Km
ρ	node density ($\rho = N/\pi R^2$)	0.0016 node/m^2
C	width of the square cell	100 m
R_c	communication range of a node	50 m
b	width of the forwarding beam	150 m
L	number of keys bound to a cell	10
m	number of MACs carried in a report	5
s	length of each MAC in bytes	4

Table 1: Notations and default parameter settings

In the analysis, we consider a circular terrain with a radius of R, over which N sensors are uniformly spread at random. The sink is located at the center of the terrain, defined as the origin in the $2D$ coordinate space. Our analysis can be applied to other forms of terrain shape, such as rectangle, and sink location. However, the presentation will be more involved. Table 1 summarizes the notations used hereinafter, and the default parameter settings in our numeric evaluations.

4.1 Filtering Effectiveness Analysis

We first consider a base setting where there is one compromised node (or equivalently, non-colluding compromised nodes), and analyze the filtering performance of LBLD using two metrics: (1) *detection ratio*: how much chance does LBLD have in filtering a fabricated report, and (2) *filtering position*: how far has a fabricated report traversed when it is filtered. In other words, we are interested in not only how many fabricated reports are filtered, but also where they are filtered, to eliminate false alarms and save network resource (e.g., energy).

Let node Z be the compromised node, with a distance of d_0 to the sink. A fabricated report injected by node Z is forwarded along a multihop path to the sink, denoted by $Z \rightarrow A_1 \rightarrow \cdots \rightarrow A_h \rightarrow Sink$, in which the h nodes A_i $(1 \leq i \leq h)$ are intermediate forwarding nodes.

4.1.1 Detection Ratio

Because the compromised node has at most one key for any cell, it has to forge at least $m - 1$ MACs, which will be detected either en-route or at the sink. This leads to a detection ratio of $1 - 1/2^{8s(m-1)}$. Given a secure hash function in generating the MACs, and a security setting with reasonable number and length of MACs, the brute-force MAC fabrication has almost negligible chances to succeed.

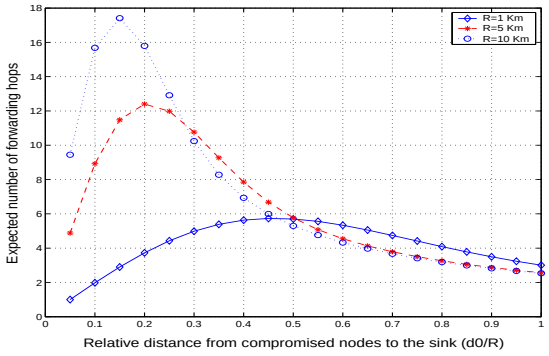

Figure 3: LBLD can quickly filter fabricated reports en-route, and its filtering power scales well as the network size increases.

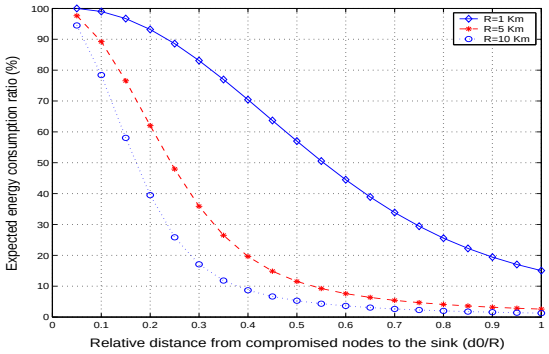

Figure 4: LBLD significantly saves energy by filtering fabricated reports en-route, especially in large-scale networks.

4.1.2 Filtering Position

LBLD can quickly filter the fabricated reports en-route by accumulating the filtering power along the forwarding path. This is shown by the following theorem (proof in Appendix).

Theorem 1 *The filtering position h', defined as the expected number of hops that a fabricated report can traverse, is upper bounded as:*

$$h' \leq 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} (1 - \frac{(m-1)(d_0 - jR_c)}{RL}) \tag{4}$$

We illustrate the above results in Figure 3, which plots the filtering position versus the compromised node's location, specified by its relative distance to the sink. In this figure, we fix the node density and vary the terrain radius R from 1 Km to 10 Km, and the node population N from 4K to 400K, respectively

(see Table 1 for other parameter settings). We can see that in a 1Km-radius network, a forged report traverses only 4.2 hops on average, and 6 hops at most. In contrast, without LBLD, a forged report can traverse as many as 20 hops. Moreover, when the terrain radius increases from 1Km to 10Km, leading to an 100-fold increase in node population, the average distance traversed by a forged report only doubles (from 4.2 hops to 7.2 hops), while the worst-case distance only triples (from 6 hops to 18 hops). This shows that the filtering power of LBLD scales very well when the network size increases.

4.1.3 Energy Saving

The early dropping of forged reports leads to significant energy savings, especially in large-scale sensor networks. Assuming that all nodes in the network use the same transmission power, we plot in Figure 4 the energy consumption ratio between the LBLD-protected paths and the unprotected paths, i.e., h'/h. The figure shows that LBLD can save energy by a ratio of 43.7% in a 1Km-radius network, and 81.3% in a 10Km-radius network. Such an increase in the energy saving ratio is because the filtering position in LBLD increases much slower than the network size.

Figure 4 also shows that the energy savings of LBLD also depends on the compromised node's location. This is because each node picks up its verifiable cells in a probability proportional to its distances to the sink (Equation 3). As a result, when the compromised node is further away from the sink, the downstream nodes along the forwarding path have larger chances to detect the fabricated reports, which are dropped more quickly.

4.2 Resiliency Analysis

Now we analyze the resiliency of LBLD to an increasing number of compromised nodes. We consider a general case where the attacker compromises N_c nodes and fabricates reports on bogus events "happening" in cell (X, Y). The results show that the security protection offered by LBLD degrades gracefully, rather than completely breaking down in the entire network as in existing designs [28, 32].

4.2.1 Graceful Performance Degradation

The attacker cannot arbitrarily abuse the keys due to their location-binding nature. To fabricate reports without being detected, the attacker must collect m distinct keys bound to cell (X, Y). We term this as *cell compromise*. Even in such cases, the attacker cannot use these keys to forge events in other cells. Thus the fabricated reports reveal important diagnostic information to the sink. The sink can quarantine the compromised cells by informing the nodes not to forward any reports for them. This way, the sink may lose monitoring capability in a few cells, but the rest of the network is still protected by LBLD.

The location-based key assignment ensures that the compromised cells represent only a tiny fraction of the terrain. When the compromised nodes are

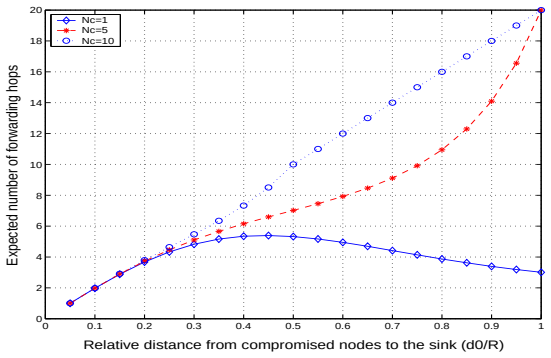

Figure 5: The performance of LBLD degrades gracefully even in the worst-case scenarios.

randomly distributed, the simulation results in Section 5 show that the chance of cell compromise decreases *exponentially* with respect to m. Each additional MAC reduces the probability of comprosing a cell by 10 to 100 fold.

Below we consider the *worst-case* scenarios where all N_c compromised nodes are local neighbors, with a distance of d_0 to the sink. Because neighboring nodes have largest correlation in their keys, the attacker has largest chance in compromising a cell. For example, when $N_c > m$, the attacker can compromise the cell where these nodes reside, and fabricate events in this cell without being detected. In addition, he may compromise a few remote cells, but LBLD limits the compromised remote cells within the upstream region of the compromised nodes. Based on Equation 3 we know that the attacker can collect $\frac{d_0 N_c}{R}$ keys of a remote cell on average. When the attacker forges events in a remote cell, the degradation of LBLD's filtering power is characterized in Theorem 2 (proof in Appendix).

Theorem 2 *With N_c neighboring compromised nodes, the filtering position h' is upper bounded as:*

$$h' \leq 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} (1 - \frac{(mR - d_0 N_c)(d_0 - jR_c)}{R^2 L}) \tag{5}$$

Figure 5 illustrates the above graceful performance degradation in the worst-case scenarios. In this figure, we fix the node population as 4K and terrain radius as 1Km, and gradually increase N_c, the number of compromised nodes. The figure shows that the expected number of forwarding hops for fabricated reports increases only slightly as more nodes are compromised. For example, when N_c increases from 1 to 5, on average LBLD can still filter fabricated reports in 7.3 hops, leading to 27% energy savings. We emphasize that the above is a worst-case analysis, and LBLD is much more effective in average cases, which we will show in Section 5 using simulations.

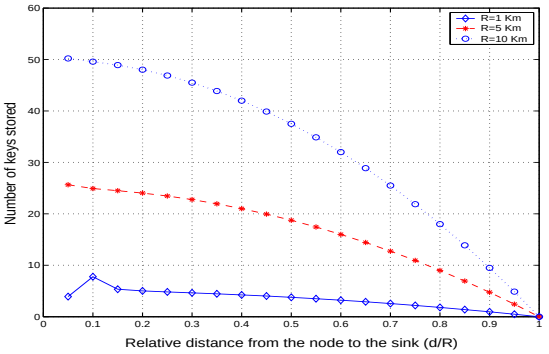

Figure 6: Each node stores only a small number of keys, and the key storage overhead scales well in large networks.

4.3 Overhead Analysis

Now we analyze the overhead of LBLD in terms of key storage, computation, and communication overhead.

4.3.1 Key Storage

In LBLD, each node stores one keys for each sensing cell and a few remote verifiable cells. The number of sensing cells is a constant, decided by the sensing range and the cell size. Thus we count only the number of keys for remote verifiable cells. Based on Equation 3, we can show the key storage overhead in the following theorem (proof in Appendix).

Theorem 3 *The number of keys stored by a node is:*

$$N_{key} \approx \frac{d(R^2 - d^2)}{2RC^2} \times \arcsin \frac{b}{max(b, 2d)} = O(\frac{bR}{C^2}) \tag{6}$$

where d is the node's distance to the sink.

Despite its strong filtering power, LBLD only requires the nodes to store a small number of keys. As shown in Figure 6, when 4K nodes are spread over a 1Km-radius terrain, each node stores only 3.35 keys (not including the constant number of sensing cell keys) on average, and 8 keys at most. Note that the terrain is divided into roughly 300 cells in this setting. This clearly demonstrates the efficiency of LBLD. The key storage overhead is also location-dependent. A node closer to the sink tends to store more keys, mainly because it has a much larger upstream region. Moreover, the key storage overhead scales well because it increases almost linearly with the terrain radius, i.e., $O(\sqrt{N})$ given a fixed node density. Even in a network with 400K nodes and 30K cells, each node stores only 32.1 keys on average, and 50 keys at most, which is still within the resource limitation of existing sensor hardware.

4.3.2 Computation and Communication

LBLD uses light-weight one-way function as its cryptography primitive, thus computationally efficient. The communication overhead of LBLD mainly comes from the MACs carried in the reports. Existing MAC compression techniques, such as Bloom filters [28], can be applied to reduce the communication overhead. However, we do not further explore on this issue due to space limits.

4.4 Security Analysis

While LBLD is designed to address event fabrication attacks, it may be abused by the attacker to launch other attacks. We provide below a security analysis on two types of such attacks.

4.4.1 Report Disruption Attacks

The attacker may abuse LBLD to disrupt the generation of legitimate reports on real events. Specifically, the attacker may launch the following attacks: 1) *MAC falsification attacks* in which a compromised node announces an incorrect MAC to its neighbors; 2) *impersonation attacks* in which a compromised node impersonates another legitimate node; and 3) *Sybil attacks* [6, 15] in which a compromised node presents multiple identities and announces one incorrect MAC in each identity. As a result, the final report may be poisoned by such incorrect MACs, and dropped in delivery or finally rejected by the sink.

Note that a compromised node cannot launch the above attacks against a remote area due to its limited transmission range. Instead, it has to be physically close the event's location. A local authentication mechanism, e.g., pairwise keys [9] and μTESLA [16], or Sybil defense mechanism [15] can limit the damage of such attacks: As long as we ensure that each node can announce only one MAC, the chance that a legitimate report is properly generated is large. Also, when the sensing range of the nodes is larger than half of their communication range, the detecting nodes of an event may reside in different communication neighborhoods. The legitimate nodes one-hop away from the compromised nodes can still properly generate the report.

4.4.2 Sensor Relocation Attacks

The attacker may physically relocate some nodes from their original locations to a new one. When a real event happens nearby the new location, these nodes may generate an incorrect report using their original locations. There are two cases: a) The attacker has already compromised the relocated nodes. Thus the sensor relocation attacks do not incur additional damage to LBLD, because the attacker are already able to control the compromised nodes to fabricate any reports. b) The attacker has not compromised the relocated nodes. In such cases, sensor relocation attacks can be defeated by local authentication mechanisms, because the relocated nodes cannot establish trust with their new neighbors, and hence their reports will not be forwarded.

4.5 Impact of Design Parameters

Finally we analyze the impact of design parameters.

The cell size, C, affects the tradeoff between key storage and protection granularity, but it does not change the filtering power. When C increases, the number of keys stored by each node decreases in proportion to $1/C^2$ (Equation 6), and it becomes harder for the attacker to collect enough keys of a cell. However, when the attacker happens to have collected enough keys, e.g., through compromising multiple local nodes, he can fabricate events in a larger area. The cell size should not exceed the application requirement on the location accuracy of event reports.

The number of keys bound to a cell, L, impacts filtering power and generation of legitimate reports. As shown in Equation 4, a smaller value of L leads to larger filtering power; however, it also increases the chance that two neighboring nodes are preloaded with the same master secret. In such cases, they can contribute only one distinct MAC when a real event occurs.

The number of MACs carried in a report, m, trades off communication overhead for security protection. Clearly, the more MACs each report carries, the stronger protection LBLD can provide. However, it comes at the price of increased communication overhead. Typically, a value between 5 and 10 provides a good tradeoff point between the two extremes.

The choice of b, the width of forwarding beams, does not affect the filtering power. However, it impacts both storage overhead and delivery ratio of legitimate reports. The key storage overhead linearly with respect to b (Equation 6), but a large beam width can improve the report delivery ratio by avoiding unnecessary dropping of legitimate reports when they traverse outside the forwarding beam, e.g., due to unexpected node failures. We will further evaluate this in the next Section.

5 Performance Evaluation

In this section, we evaluate the performance of our design through implementation and simulations.

5.1 Simulation Results

We use simulations to study such performance aspects that cannot be evaluated easily through analysis or experiments. Specifically, we evaluate the resiliency of LBLD under random node compromise, and validate the practicality of the beam model on geographic forwarding.

5.1.1 Resiliency to Random Node Compromise

Given that we have analyzed the worst-case resiliency of LBLD when multiple compromised nodes are local neighbors, we are interested to use simulations to study its average-case performance, when the attacker *randomly* compromise

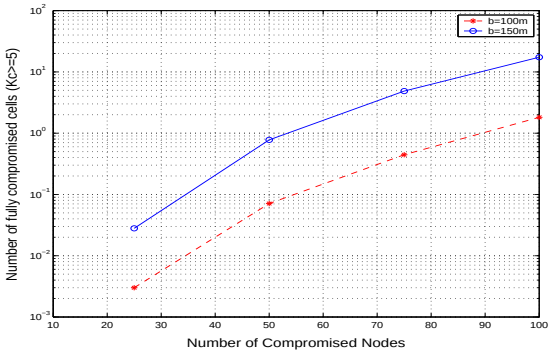

Figure 7: The attacker can hardly collect enough keys bound to a cell when the compromised nodes are scattered in the network.

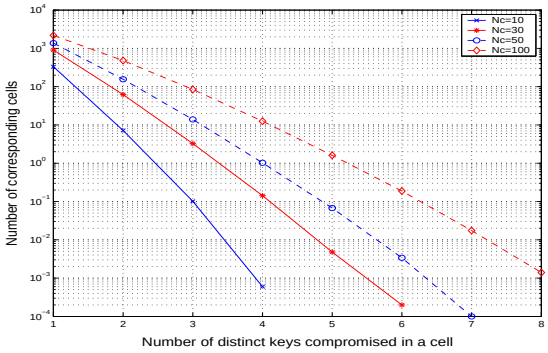

Figure 8: The difficulty to obtain more distinct keys of a cell increases exponentially.

multiple nodes in the network. For this purpose, we developed our own simulation platform using Parsec [1], mainly because the existing simulators scale poorly when simulating a large number of nodes. Our simulator implemented the basic geographic forwarding [13] and the LBLD protocol stack. We simulated rectangular terrains to complement our circular terrain based analysis. The parameter settings are similar to the default values in Table 1, unless explicitly mentioned. Our simulation results show that LBLD is highly resilient to random node compromises.

We first simulate how many distinct keys of a particular cell the attacker can obtain by compromising multiple nodes, which is the key factor in evaluating the resiliency of LBLD. In the simulations, 30K nodes are spread over a 5Km$\times$ 5Km field, divided into 100m$\times$100m cells. The sink is located at the center of the field. We vary the beam width b with 100m and 150m, and gradually increase the number of randomly chosen compromised nodes from 10 to 100. Each simulation setting is repeated 1000 times with different random network topology and distribution of compromised nodes.

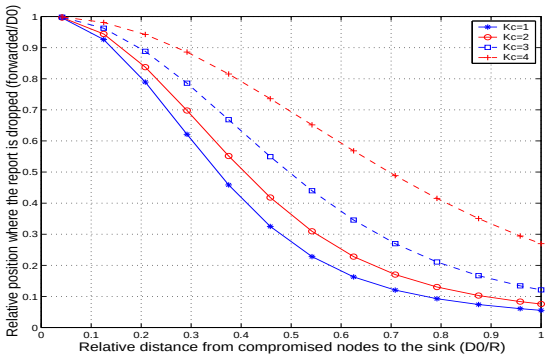

Figure 9: The filtering power of LBLD degrades gracefully when the attacker has collected more keys of a cell.

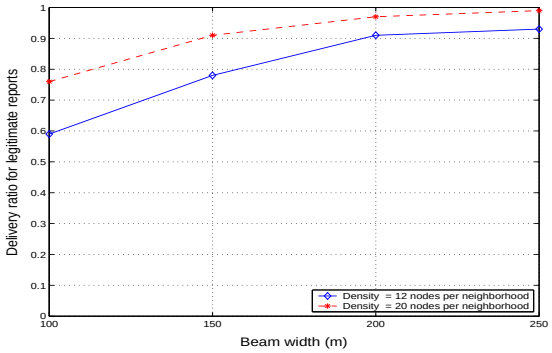

Figure 10: With a moderate beam width, most legitimate reports can be successfully delivered to the sink.

The simulation results show that even with combining all keys stored at all compromised nodes, it is still quite difficult for the attacker to obtain enough keys bound to a same cell. This is illustrated in Figure 7, which plots the number of fully compromised cells versus the number of compromised nodes. Because we carry 5 MACs in each report, the attacker can fully compromise a cell when it has collected 5 distinct keys bound to the cell. When the beam width is chosen as 150m, 100 compromised nodes only lead to the compromise of 17 cells, or 0.68% of the entire terrain. A decrease of the beam width to 100m further constraints the damage as only 1.8 fully compromised cells on average.

Figure 8 provides a more detailed view on the aggregate effects of combining the keys at multiple compromised nodes. We fix the number of compromised nodes as 100 in these simulations. The figure plots, in a log-linear manner, the histogram of cells versus their level of damages, i.e., how many distinct keys the attacker has collected. The X axis is the number of distinct keys exposed to the attacker, and the Y axis is the total number of cells with a corresponding

number of keys being exposed. We can see that the chances of collecting q distinct keys of a cell decreases exponentially with respect to q. Therefore, by carrying more MACs in the reports, we can significant enhance the resiliency of LBLD to even a very large set of compromised nodes.

Next we simulate how the filtering power of LBLD degrades when the attacker happens to have multiple keys of a cell. In the simulations, we vary the location where the fabricated reports is injected, from adjacent to sink to the network edge. Since each report carries only 5 MACs, there is no needs to simulate the cases when the compromised nodes have 5 or more keys. The result is shown in Figure 9. We can see that with each additional compromised key, the decrease of filtering power is only marginal, leading to graceful performance degradation.

5.1.2 Validity of Beam Model

Now we verify the validity of the beam forwarding model used in estimating the upstream regions. In particular we want to confirm that the legitimate reports would indeed be forwarded to sink without being accidentally dropped by a node outside the forwarding beam. For this purpose, we vary the beam width b with 100m, 150m, 200m, and 250m, and simulate different node density from 12 to 20 nodes per communication neighborhood.

The delivery ratio of legitimate reports is plotted in Figure 10, which shows that with a moderate beam width of 200m, the delivery ratio can be as high as 99.2% in a dense network, and 90.6% in a relatively sparse network. Note that the transmission range of each node is 50m in the simulations. Thus the beam width is roughly four-hop communication range. Clearly the delivery ratio depends on the relationship between the beam width and the node density. The beam width should be set based on the expected node density. With decreased node density, the beam width should increase accordingly to ensure a high delivery ratio.

5.2 Implementation and Measurement Results

We implemented the LBLD design on MICA2 motes developed by X-Bow [3]. These tiny computing devices are equipped with an 8-bit 4MHz microcontoller running a microthread operating system, called TinyOS [2], from its internal flash memory. The memory size available at each node is limited: 128KB of program memory and 4KB of data memory. These stringent resource constraints clearly require a compact implementation that can fit into the underlying hardware platform.

We implemented a cryptographic primitive of secure hash function based on a block cipher using RC5 algorithms [18]. This module facilitates the derivation of location-binding keys, as well as generating and verifying MACs. We also implemented a generic wireless communication module to exceed the packet size limit of 29 bytes in TinyOS *GenericComm* interface. It directly reads

Module	ROM	RAM
Bootstrapping	236	58
Report Generation	2820	225
Filtering	106	40
Radio Stack	4130	114
RC5-Crypto	646	128
Others(Timer, Sensing Drivers)	1420	100
Total	9358	665

Table 2: Code size breakdown (in bytes) in MICA2 Platform

and writes the buffer associated with the low-level radio device, and thus can transmit packets of any length.

5.2.1 Code Size

Table 2 shows a breakdown of the implementation codes size on the MICA2 platform. The LBLD protocol stack (i.e., bootstrapping, report generation, and filtering) consumes around 3.2K bytes in ROM and 323 bytes in RAM. Together with the communication module, cryptography module, timer and sensor drivers, the entire system consumes 9.4K bytes in ROM and 0.67K bytes in RAM, or 7.3% and 16.6% in percentages for ROM and RAM, respectively.

5.2.2 Execution Time

Our measurement results show that, given a grid of 100×100 cells, it takes a MICA2 mote 2.8 seconds to derive the cell keys in the bootstrapping phase. The master key is permanently erased afterwards, posing high time constraints for an attacker to compromise the critical master key. The MAC generation and verification is also fast: 10 ms to generate or verify one MAC for a 24-bytes report.

6 Discussion

In this section we comment on several design issues and identify future research directions.

Secure Localization Our location-based security design relies on a secure localization protocol, so that the sensor nodes can securely obtains their locations within certain accuracy. In fact, secure localization is required in most monitoring applications to identify the correct location of events. We plan to devise security solutions for sensor localization protocols [20, 30] in the future.

Asynchronous Deployment The deployment of a large sensor network may take long time (e.g., a few hours). After the initial deployment, we may still refill new nodes to replace the failed or out-of-battery ones. In such cases, the location-aware anchor nodes should be deployed before or together with normal

nodes to ensure fast localization, so that each node can bootstrap and erase the master secrets on time, regardless of the global deployment status. We can further protect the master secrets by setting a timer at each newly deployed node, and erasing the master secret when the timer fires, even though it has not been bootstrapped. This may lead some nodes to be useless; however, given the high density, the network can still function well as a whole.

Terrain Shape The beam forwarding model, used in upstream region estimation, works well with convex terrains. However, it may not hold in concave terrains (e.g., half-moon shapes). In such cases, a node should use different strategies, such as a uniform one, in selecting its verifiable cells. We will further investigate this issue in the future.

Node Density Topology control protocols [29, 25] are commonly used to prolong the sensor network lifetime by turning redundant nodes into sleeping. However, our design requires multiple nodes to jointly generate a report. Thus, a sleeping node should leave its sensing module on, while turning off the major energy-consumer hardwares such as radio. Once an event happens, nearby nodes wake up, triggered by the sensing module, and collaborate in generating the reports. This way, we can achieve both energy efficiency and high *sensing* density.

Routing The upstream region estimation scheme in LBLD is designed to work with geographic routing protocols. Yet several non-geographic sensor routing protocols, such as Directed Diffusion [11] and GRAB [26], are also shown to fit well with the beam forwarding model, and thus can potentially work with LBLD. However, a careful investigation is needed and we leave it for future research.

Key Update and Revocation One limitation of our current design is that it does not include key update or revocation. In future research, we plan to address this problem by binding keys to a spatial-temporal space, i.e., with both time and location. Thus we can revoke the current keys by providing forward security in the temporal key chain, e.g., through a reversed hash chain.

7 Conclusion

Node compromise presents severe security threats in sensor networks. To this end, we have presented the design of LBLD that pursues a location-based approach in addressing report fabrication attacks launched by compromised nodes. One salient feature of LBLD is that its security protection degrades gracefully to an increasing number of compromised nodes. Such resiliency is mainly achieved through two novel techniques: location-binding keys and location-based key assignment. In essence, LBLD provides a desirable balance between *secret sharing* and *secret separation*. It enables the sensor nodes to collaborate in securing the network by sharing symmetric keys, yet limits the scope and usage of individual keys, and exposes only a minimum set of keys to each node. We believe that such a location-based design approach can be applied to tackle many security problems, such as secure routing and DoS quarantine, to defend against

compromised nodes in sensor networks.

References

[1] PARSEC: Parallel simulation environment for complex systems. http://pcl.cs.ucla.edu/projects/parsec/.

[2] TinyOS operating system. http://millennium.berkeley.edu.

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

[4] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE SSP*, 2003.

[5] J. Deng, R. Han, and S. Mishra. Security support for in-network processing in wireless sensor networks. In *ACM SASN*, 2003.

[6] J. Douceur. The Sybil Attack. In *IPTPS02 Workshop*, 2002.

[7] 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.

[8] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM CCS*, 2003.

[9] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *ACM CCS*, 2002.

[10] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *IEEE INFOCOM*, 2004.

[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] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *ACM MOBICOM*, 2000.

[14] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *ACM CCS*, 2003.

[15] J. Newsome, R. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *ACM IPSN*, 2004.

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

[17] B. Pryzdatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SENSYS*, 2003.

[18] R. Rivest. The RC5 encryption algorithm. In *Workshop on Fast Software Encryption*, 1995.

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

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

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

[22] 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.

[23] G. Tsudik. Message authentication with one-way hash functions. *ACM CCR*, 22(5):29–38, 1992.

[24] A. Wood and J. Stankovic. Denial of service in sensor networks. *IEEE Computer*, October 2002.

[25] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MOBICOM*, 2001.

[26] F. Ye, S. Lu, and L. Zhang. GRAdient Broadcast: A robust data delivery protocol for large scale sensor networks. *ACM WINET*, March 2005.

[27] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *ACM MOIBCOM*, 2002.

[28] F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. In *INFOCOM*, 2004.

[29] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS:a robust energy conserving protocol for long-lived sensor networks. In *IEEE ICDCS*, 2003.

[30] 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.

[31] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanism for large-scale distributed sensor networks. In *ACM CCS*, 2003.

[32] 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 SSP*, 2004.

8 Appendix

PROOF of Theorem 1 Let the forwarding path of the fabricated report be $Z \rightarrow A_1 \rightarrow \cdots \rightarrow A_h \rightarrow Sink$. The geographic distance from the compromised node Z to the sink is denoted by d_0, while the distance from an intermediate forwarding node A_i to the sink is denoted by d_i. Since the maximum transmission range of a node is R_c, we know that $d_i \geq d_0 - iR_c$.

Consider the action taken by node A_i after it receives the fabricated report. Node A_i drops the report if the claimed event's location is outside its upstream region. Otherwise, node A_i has a probability of d_i/R (Equation 3)to have a key bound to the event's cell, thus able to verify the report. On the other hand, the compromised node has to forge at least $m-1$ MACs in the report. Therefore, the probability that node A_i drops the report, given then it has received the report, is at least:

$$P_i = \frac{(m-1)}{L} \times \frac{d_i}{R} \geq \frac{(m-1)(d_0 - iR_c)}{RL}$$

Because each forwarding node performs the same checking, the entire path collectively exhibits strong filtering power. The filtering position h', defined as the expected number of hops that the fabricated report can traverse, can be derived as follows.

$$\begin{aligned} h' &= 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} (1 - P_j) \\ &\leq 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} (1 - \frac{(m-1)(d_0 - jR_c)}{RL}) \end{aligned}$$ □

PROOF of Theorem 2 The proof is similar to the previous one. When the attacker has compromised N_c node in a local neighborhood, he can collect $\frac{d_0 N_c}{R}$ keys bound to a remote cell on average, where d_0 is the distance from these compromised nodes to the sink. Thus he needs to forge $m - \frac{d_0 N_c}{R}$ MACs in the report. Accordingly, the probability that node A_i drops the report becomes:

$$P_i = \frac{(m - d_0 N_c/R)}{L} \times \frac{d_i}{R} \geq \frac{(mR - d_0 N_c)(d_0 - iR_c)}{R^2 L}$$

Similarly, the filtering position is upper bounded as:

$$\begin{aligned} h' &= 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} (1 - P_j) \\ &\leq 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} (1 - \frac{(mR - d_0 N_c)(d_0 - jR_c)}{R^2 L}) \end{aligned}$$ □

PROOF of Theorem 3 Consider a node with a distance of d to the sink. Let Γ denote its upstream region, as defined in Section 3.2 (See Figure 2 for a graphical illustration). Recall that we have derived the spanning angle of Γ, denoted by α, in Equation 2.

Based on Equation 3, we know that the node stores one key for each cell in Γ with a probability of $\frac{d}{R}$. Thus, the number of verifiable cell keys stored by the node is proportional to the number of cells within Γ. That is,

$$\begin{aligned}
N_{key} &= \sum_{Cell\,(X_i,Y_j)\in\Gamma} \frac{d}{R} \\
&\approx \frac{d}{RC^2} \iint_{\Gamma} r\,\mathrm{d}r\,\mathrm{d}\theta \\
&= \frac{\beta d}{RC^2} \int_0^{\alpha}\int_d^{R} r\,\mathrm{d}r\,\mathrm{d}\theta \\
&= \frac{\alpha d(R^2-d^2)}{2RC^2} \\
&= \frac{d(R^2-d^2)}{2RC^2} \times \arcsin\frac{b}{max(b,2d)} \\
&= O(\frac{bR}{C^2}) \qquad \square
\end{aligned}$$