Toward Resilient Security in Wireless Sensor Networks*

Hao Yang*, Fan Ye†, Yuan Yuan‡, Songwu Lu*, William Arbaugh‡

*UCLA Computer Science
Los Angeles, CA 90095
{hyang,slu}@cs.ucla.edu

†IBM T.J. Watson
Hawthorne, NY 10532
fanye@us.ibm.com

‡Dept. of Computer Science
University of Maryland, MD 20742
{yuanyuan,waa}@cs.umd.edu

ABSTRACT

Node compromise poses severe security threats in wireless sensor networks. Unfortunately, existing security designs can address only a small, fixed threshold number of compromised nodes; the security protection completely breaks down when the threshold is exceeded. In this paper, we seek to overcome the threshold limitation and achieve resiliency against an increasing number of compromised nodes. To this end, we propose a novel location-based approach in which 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 scope for which individual keys can be (mis)used, thus limiting the damages caused by a collection of compromised nodes. We illustrate this approach through the problem of report fabrication attacks, in which the compromised nodes forge non-existent events. We evaluate our design through extensive analysis, implementation and simulations, and demonstrate its graceful performance degradation in the presence of an increasing number of compromised nodes.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Network**]: Security and protection; C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Design, Security

Keywords

Wireless sensor networks, location-based security, resiliency, node compromise, en-route filtering, key distribution

*This research has been supported in part by DARPA SensIT program under contract number DABT63-99-1-0010 and in part by NSF CAREER ANI-0093484.

MobiHoc'05, May 25–27, 2005, Urbana-Champaign, Illinois, USA.

1. INTRODUCTION

Wireless sensor networks are ideal candidates to monitor the environment in 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 vast terrain to detect events of interest (e.g., enemy vehicles, forest fires), and deliver data reports over multihop wireless paths to the user. Security is essential for these mission-critical applications to work 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 compromised nodes and use them to launch *insider* attacks. This threat is aggravated as the adversary compromises more nodes and secret keys. Unfortunately, most existing security designs [2, 5, 12, 23, 28] 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 constraint that prevents the attacker from compromising more than the threshold number of nodes.

In this paper, 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 location-based security approach through two techniques: *location-binding keys* and *location-based key assignment*. In this approach, we bind symmetric secret keys to geographic locations, as opposed to sensor nodes, and assign such location-binding keys to sensor nodes based on their deployed locations. We illustrate these concepts in the context of *report fabrication attacks*, where the compromised nodes forge non-existent events that cause both false alarms and network resource waste (see more details in Section 2). Our design, a Location-Based Resilient Security (LBRS) solution, demonstrates that such a location-based approach can effectively limit the damage caused by even a large collection of compromised nodes.

In LBRS, 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 detect fabricated reports, we require that a real event be endorsed through multiple keys bound to the specific location of the event. An attacker that has compromised multiple nodes may obtain keys bound to different cells, but he cannot combine such keys to fabricate any event

without being detected. To limit the damage of network 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 constrains 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 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 show that LBRS is resilient, efficient, and scalable. For example, in a network of 4000 nodes with each node storing less than 8 keys, LBRS can drop fabricated reports after 4.2 hops on average. When the adversary has compromised 100 nodes, LBRS can still prevent false alarms in 99% of the field.

The rest of this paper is organized as follows. We overview the event fabrication attack problem and the general en-route filtering framework in Section 2, and examine the (in)resiliency of existing solutions in Section 3. We describe our location-based security solution in Section 4, and analyze its resiliency and overheads in Section 5. We further evaluate our design through simulation results in Section 6 and testbed implementation in Section 7. We discuss several design issues in Section 8 and compare to the literature in Section 9. Finally, we conclude the paper in Section 10.

2. BACKGROUND

In this section, we first describe the problem of report fabrication attacks in sensor networks, then review the general en-route filtering framework as a countermeasure.

2.1 Report Fabrication Attacks

We consider a large-scale sensor network that monitors a vast geographic terrain using a large number of static sensor nodes. An approximate estimation on the size and shape of the terrain being monitored is known *a priori*. Each sensor node is battery-powered and has limited sensing, computation and wireless communication capabilities. The sensor deployment is dense enough to support fine-grained collaborative sensing and provide robustness against node failures. For simplicity, we assume that the node distribution is uniform. Once deployed, each node can obtain its geographic location via a localization scheme [18, 26].

In a sensor network that serves mission-critical applications such as battlefield surveillance and forest fire monitoring, prompt detection and reporting of each relevant event in the field is critical. When an event occurs, the detecting nodes generate a report message and deliver it over multihop wireless channels to the *sink*, the data collection unit that is typically a resource-abundant computer. In our model, the sink is static and its location is known when sensors are deployed. Once the sink receives an event report, response actions such as sending personnel and facilities to the event's location, can be taken subsequently.

Unfortunately, the above event detection operations can be severely disrupted by *report fabrication attacks*. In such attacks, the adversary compromises a single or multiple nodes, then uses them to inject forged sensing reports that describe *non-existent* events. The compromised node(s) can pretend to have "detected" a nearby event or "forwarded" a report originated from a remote location. Therefore, the forged events could "appear" not only where nodes are compromised, but also at *arbitrary* locations. Such bogus reports can deceive the user into wrong decisions and result in the failure of mission-critical applications. They can also induce congestion and wireless contention, and waste a significant amount of network resources (e.g., energy and bandwidth), along data delivery paths. In the worst case, a large number of forged reports can disrupt the delivery of legitimate reports and deplete the energy of forwarding nodes.

In this paper, we consider the following *threat* model. The attacker may compromise multiple sensor nodes in the network, and we do not impose any upper bound on the number of compromised nodes. However, the attacker cannot compromise the sink, which is typically resourceful and well-protected [15]. Once a sensor node is compromised, all secret keys, data, and code stored on it are exposed to the attacker. The attacker can load a compromised node with secret keys obtained from other nodes. We term this as *collusion* among compromised nodes. The compromised nodes can launch many other attacks, such as dropping legitimate reports, to disrupt the network operations. However, these threats are addressed in other related work [20, 21] and are not the focus of this paper. We will study the impact of a few of them upon our design in Section 5. We also assume that the attacker cannot successfully compromise a node during the short deployment phase, i.e., the interval of tens of seconds when each sensor bootstraps itself (including obtaining its location and deriving a few keys). Some existing work [1, 27] has made similar assumptions and argued that such attacks can indeed be prevented in real-life scenarios when appropriate network planning and deployment keep away attackers during the bootstrapping process. We will revisit this aspect in Section 8.

2.2 General En-route Filtering Framework

We follow the general en-route filtering solution framework [23, 28] in defending against report fabrication attacks. The framework has three components that work in concert: report generation using Message Authentication Codes (MACs), en-route filtering, and sink verification.

To be forwarded and accepted downstream, a legitimate report must carry m ($m > 1$) distinct MACs from the sensing nodes. Each node stores a few symmetric keys and endorses any event it has observed by using its keys to generate a MAC on the report. Each key has a unique index, and the sink knows all the keys. When a real event occurs, multiple detecting nodes jointly generate a complete report with the required m MACs and the associated key indices.

The intermediate nodes detect and discard bogus reports injected by compromised nodes. When a node receives a report, it verifies the report as follows: It first checks whether the report carries m distinct MACs. It then searches its own stored keys for matched key indices. When a match is found, it checks whether the carried MAC is the same as the MAC it computes via its locally stored key. It drops the report when any of these checks fails. Otherwise (i.e., it does not

have any of the keys or the MACs are correct), it forwards the report as usual. Even though the filtering power (i.e, the detection percentage for forged reports) at each node may be limited, the collective filtering power along the forwarding path can be significant. The more hops a forged report traverses, the higher chance it is dropped en-route. Consequently, one can effectively exploit the sheer scale of the sensor network in filtering the forged reports.

The en-route filtering performed by sensor nodes may be probabilistic in nature, thus cannot guarantee to detect and drop all forged reports. The sink serves as the final guard in rejecting any escaping ones. Because the sink knows all the keys, it can verify each MAC carried in a report. Note that there might be multiple reports for the 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.

Three designs, including Statistical En-route Filtering (SEF) [23], Interleaved Hop-by-hop Authentication (IHA) [28] and our design in this paper, are all specific instances within the above framework.

3. ON RESILIENCY OF EXISTING SOLUTIONS

In the above framework, there exists a fundamental design 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 a larger chance to detect and drop forged reports. However, to enhance the resiliency, each node should store fewer keys to minimize the damage of compromised nodes, because the attacker can abuse the subverted keys to endorse bogus reports. How to resolve this conflict and achieve both resiliency and effective en-route filtering in a scalable fashion is a fundamental security challenge for large sensor networks.

Unfortunately, none of the existing security solutions for sensor networks meets the resiliency requirement. In fact, most of them [2, 4, 5, 6, 12, 27] are not designed to address compromised nodes. In the reminder of this section, we will illustrate the resiliency problems of two solutions that are explicitly designed for report fabrication attacks: IHA [28] and SEF [23].

IHA [28] verifies the reports in a deterministic and hop-by-hop fashion. In the deployment phase, each node is pre-loaded with a unique ID and keying materials that can allow it to establish a pairwise key with another node. The nodes form multiple clusters and each cluster has at least $t + 1$ nodes, where t is a design parameter. Each cluster head discovers a path to the sink. Along the path, two nodes that are $t + 1$ hops away are *associated* by establishing a pairwise key. Upon an event, each detecting node computes two MACs, one using its key shared with the sink and the other using its pairwise key shared with its downstream associated node. The cluster head sends out a final report that carries the MACs from $t + 1$ detecting nodes. In the en-route filtering phase, each forwarding node verifies the MAC from its upstream associated node. Upon successful verification, it replaces the old MAC with a new one using its pairwise key shared with its downstream associated node. The sink performs final verification on the report.

IHA suffers from two major drawbacks in resiliency. First, the protection breaks down when more than t nodes along a path are compromised. In such cases the attacker can forge events "appearing" at arbitrary locations. Second, it relies on deterministic key sharing in that each node must know the upstream and downstream $(t + 1)$-hop neighbors and establishes pairwise keys with them. As routing paths may frequently change, e.g., to adapt to node failures, such deterministic key sharing needs to be maintained through repairing or rebuilt through routing [28], both of which can be expensive in terms of communication overheads, energy consumption and response time.

In contrast, SEF [23] filters the forged reports en-route in a probabilistic manner. In SEF, a global key pool is divided into multiple partitions, and each node is pre-loaded with a few keys randomly chosen from a single partition. When an event occurs, the detecting nodes jointly endorse the report with T MACs, each using a key in a different partition. SEF assigns keys to nodes in a way that any intermediate node is able to verify the report with certain probability. The sink can always verify every report because it knows the entire key pool. As a result, most of the forged reports are quickly dropped by the forwarding nodes, and the few escaping ones are further rejected at the sink.

SEF suffers from the threshold drawback similar to IHA. Its protection breaks down when the attacker has obtained keys in T partitions. Because each node stores keys from one partition, an attacker who compromises a threshold number[1] (e.g., $2T$) of nodes can subvert SEF and freely forge any report with an almost-one probability. Nevertheless, SEF is robust against node failures and routing path changes. The verification can be performed by any forwarding node and does not rely on any deterministic secret sharing among nodes. Therefore, no maintenance is needed when nodes fail along the forwarding path or the routing paths change.

In summary, the existing solutions are not resilient against an unbounded number of compromised nodes. The fundamental problem is that, both the credential (i.e., MAC) generation and its verification rely on the same secrets in symmetric-key based designs, which are commonly used for resource-constrained sensor networks. Strong verification power requires more sharing of secrets among the nodes, but high degree of resiliency demands the opposite, i.e., more separation of secrets. For the security solution to be effective yet resilient in large-scale sensor networks, secret sharing and secret separation must be well balanced.

4. DESIGN

In this section, we present the design of our Location-Based Resilient Security solution (LBRS) for report fabrication attacks. LBRS follows the general en-route filtering framework described in Section 2.2, yet achieves resiliency against both node compromise and node failure through two novel techniques: *location-binding key generation* and *location-guided key selection*.

4.1 Overall Operations

The overall operations of LBRS are as follows. As shown in Figure 1, we divide the terrain into a geographic grid and bind multiple keys to each cell on it. We term such keys as *location-binding keys*. Within the keys bound to one cell,

[1]The threshold for SEF depends on another design parameter, the total number of partitions in the key pool.

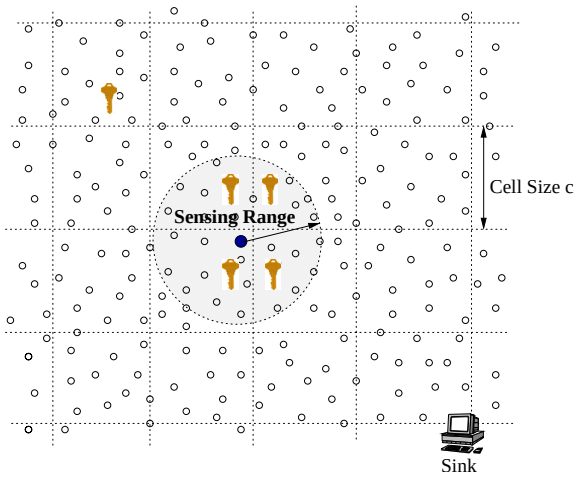

Figure 1: Each square cell on the geographic grid is associated with multiple keys. Each node stores a few local and remote cell keys based on its own location.

each of them is associated with and identified by an index, an integer between 1 and L. We assign these location-binding keys to nodes based on their deployed locations. Specifically, each node stores two types of keys. The first type is for the local cells within its sensing range, called *sensing cells*[2]. Each node stores *one* key for each of its sensing cells. Such keys are used to endorse events detected in those cells. The second type is for a few randomly chosen remote cells, called *verifiable cells*. Each node also stores *one* key for each of its verifiable cells. Such keys are used to verify events claimed to happen in those cells. As we shall see in Section 4.3, the selection of remote verifiable cells is guided by the location information of the node itself and the sink.

A legitimate report carries m distinct MACs, jointly generated by the detecting nodes using the keys bound to the event's cell. Specifically, upon an event, the detecting nodes first reach agreement on the event description, including the event's location, through techniques such as [21]. Each node then 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. Each node also records all such tuples announced by its neighbors, and constructs a complete report after it has received m distinct MAC tuples. To avoid duplicate reports, nodes overhear the wireless channel. Each node sets a random timer, and the node that first fires the timer sends out its 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 endorsing the report. Otherwise, upon timeout, it sends out its own report that carries m distinct MAC tuples, with higher priority over the unsent ones.

Note that a legitimate node participates in report generation only when it has sensed the event by itself. Thus a compromised node cannot deceive its neighbors into endorsing a forged report. The number of MACs in a report, m, provides a tradeoff between overhead and security strength. The more MACs each report carries, the stronger protection LBRS provides, yet at the cost of increased communication overhead. We will analyze this aspect in Section 5.

When an intermediate node receives a report, it verifies the report as follows: It first checks whether the report carries m distinct MACs and indices. If not, the report is dropped. It then retrieves the event's location from the report and checks whether the location is in one of its verifiable cells. If so, it checks whether it has one of the keys whose indices are carried in the report. If it has such a key, it recomputes the MAC and compares to the carried one. If the two MACs do not match, the report is dropped. Otherwise, it forwards the report.

The sink performs final verification on the received reports. It knows all location-binding keys, thus able to verify every MAC in the report. If any of the carried MACs is incorrect, the report is rejected. This way, the sink serves as the final guard to detect and drop those forged reports that have escaped probabilistic en-route filtering.

4.2 Location-Binding Key Generation

The location-binding approach to key generation in LBRS constrains the degree to which compromised nodes can abuse their keys, and minimizes the global damage that multiple local subverted nodes can cause. To successfully forge a bogus report, the attacker must collect enough keys from a single cell, because each report must be endorsed by multiple distinct MACs using keys bound to one cell. A collection of keys from different cells are useless, i.e., they cannot be combined to endorse any reports in a meaningful way. More importantly, even when the attacker has collected enough keys from one cell, he can only fabricate reports "happening" in that particular cell. Such constraints not only reveal valuable diagnosis/traceback information to the sink, but also quarantine the damaged area in terms of false alarms. As such, the shift from traditional node-based keys to location-binding keys results in graceful performance degradation, in contrast to threshold break-down in existing solutions [23, 28], when more and more nodes are compromised. We will analyze it in the next section. Moreover, the location-binding key generation also provides high-degree resiliency to node failures, because multiple nodes exist in one cell and have the same role in endorsing the real nearby events.

In LBRS, in order to facilitate the generation of location-binding keys, the terrain is divided into a *virtual*, *pre-defined* geographic grid. Once a node is deployed, it obtains its own position and then derives its location-binding keys. To make this seemingly simple operation work, we need to address the following three issues:

- How to construct the grid without maintaining a real, physical infrastructure?
- How to derive keys based on the location information in a computationally efficient manner?
- How to enhance resiliency for key generation?

Constructing the virtual grid Unlike the conventional approach that maintains a real, physical grid infrastructure [22], we construct a virtual square grid used only 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) (e.g., the sink location, or arbitrarily specified position). Accordingly, we denote a cell by the location

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

of its center (see Figure 1), which is (X_i, Y_j) such that

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

Note that the above grid does not require any actual maintenance, which can be quite complex in the presence of node failures and incur significant communication overheads.

The cell size,C, represents a tradeoff between key storage and protection granularity. Intuitively, with larger cells, each node can store fewer keys because there are fewer cells in total. This in turn increases the difficulty for the attacker to collect enough keys (i.e., requires him to compromise more nodes). However, in this case, when the attacker has indeed obtained enough keys from one cell, he can fabricate events in a larger area. We will analyze this tradeoff in Section 5.

Deriving keys in an efficient fashion Before the deployment, we preload each node with the cell size C, the reference location (X_0, Y_0), and a master secret K^I. Once deployed, a node first obtains its geographic location through a localization scheme [26], then derives the keys during a *short bootstrapping phase* as follows. It identifies which cells are within its sensing range[3] using simple geometry calculations. For each sensing cell, the node generates a key based on the cell's location (X_i, Y_j), together with K^I, through a secure one-way function $H(\cdot)$[19]:

$$K_{X_i,Y_j} = H_{K^I}(X_i||Y_j) \quad (1)$$

where $||$ denotes concatenation. In addition, the node also selects a few remote verifiable cells, a scheme to be described shortly, and derives *one* key for each of them similarly. This ends the bootstrapping phase, and the node permanently removes the master secret K_I from its storage, similar to [27]. After that, the node can no longer derive any keys.

The above key derivation is efficient because it involves only local computation of light-weight one-way functions, without any message exchange. As a result, the bootstrapping process is very fast, and the master secret is erased before the attacker can successfully compromise any node.

Enhancing resiliency In the above description, only one key is bound to each cell. To exploit the dense sensor deployment and improve the resiliency against node compromises, we bind L distinct keys to each cell. Accordingly, there are L master secrets, and each of the L keys for a cell (X_i, Y_j) is derived from one master secret:

$$K_{X_i,Y_j,s} = H_{K_s^I}(X_i||Y_j) \quad (2)$$

where K_s^I is the s-th master secret, and s is an index ranging from 1 to L. Each node is preloaded with one of the master secrets before deployment, and it still derives only one key for each local or selected remote cell.

The number of keys bound to a cell, L, impacts filtering power and generation of legitimate reports. As shown in Equation 5 (Section 5), 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 contribute only one distinct MAC when a real event occurs.

4.3 Location-guided Key Selection

Now we describe the location-guided key selection scheme used by a node to pick up its verifiable cells and the associated keys. Compared with the popular solutions of preloading sensor nodes with secret keys before deployment [28, 23, 6, 2, 5], LBRS allows the nodes to select keys in a *location-guided* manner after deployment. The goal is to retain the desirable filtering power yet improve the resiliency by limiting the number of keys exposed to individual nodes. To achieve this, we need to address the following three issues:

- How to exploit the location knowledge in key selection?
- How to select remote cells to balance key sharing and key separation?
- How to accommodate node failures?

Leveraging location knowledge in key selection In the absence of any *a priori* information, perhaps the best strategy is to uniformly randomly select cells from the grid. However, when the sink is *static* and a geographic routing protocol [7, 9] is used in report delivery, we can exploit the location information of a node and the sink to store significantly fewer keys, while retaining the filtering power.

The idea is that if a node can estimate its *upstream region*, i.e., those remote cells whose reports it may potentially forward, it only needs to pick verifiable cells from this region, rather than the entire field. However, when the node selects verifiable cells only from its upstream region, the attacker may deliberately forge events "happening" outside this region, so that the node possesses no keys to verify the carried MACs. To prevent such attacks, each forwarding node should perform one more check in verifying a report (the original rules were described in Section 4.1): When it extracts the event's location from the report, it checks whether the location resides in its upstream region. If not, it drops the report. Otherwise, it proceeds with the probabilistic MAC verification as usual.

Randomized selection to balance key sharing and separation After a node determines its upstream region (details to be described shortly), it needs to select verifiable cells and derive keys from this region. The key issue is to balance key sharing and key separation. By storing more verifiable cell keys, a node has higher degree of key sharing with the upstream nodes, which enables more powerful filtering; however, more keys exposed to each node imply less resiliency against compromised nodes. On the other hand, high-degree key separation improves resiliency but degrades the en-route filtering performance.

We take a simple, yet carefully designed, randomized selection method as follows. For each cell in a node's upstream region, the node selects it as a verifiable cell with probability

$$P = \frac{d}{D_{max}} \quad (3)$$

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 the shape are known, D_{max} can be preloaded into the nodes.

The above method has three desirable features that enhance system resiliency. First, the verifiable cells are chosen in a randomized manner. This is because if a deterministic algorithm is used, the attacker will know which cells are not the verifiable cells of a node. Thus he can fool the node by fabricating events claimed in those locations.

Second, the selection probability is solely determined by the node's location, and all cells in the upstream region are

[3]The sensing range can be preloaded into the node, or estimated by the node after deployment.

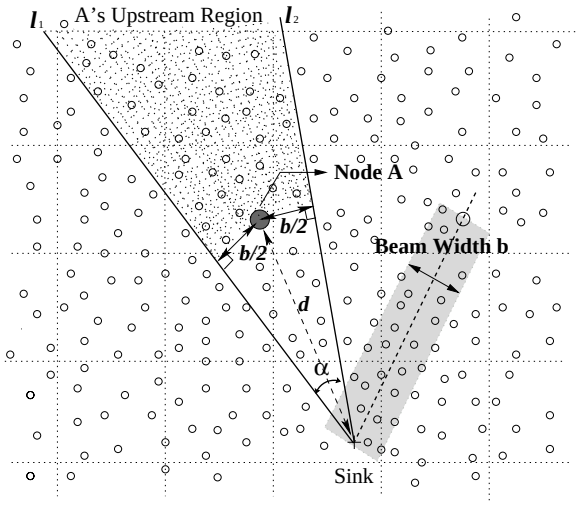

Figure 2: A report is forwarded inside a beam of width b from the source to the sink. Thus each node can estimate its upstream region.

considered equally important and chosen with equal probability. This improves the worst-case filtering performance. For any non-uniform strategy that selects some upstream cells with higher probability and others with smaller probability, the attacker can increase his chance of success by fabricating events in the unfavored cells.

Finally, the above method can filter out most forged reports at the initial several hops while minimizing the key storage. For a node further away from the sink (i.e., when d increases), it picks up verifiable cells with a higher probability; however, its upstream region shrinks, and the net effect still allows it to stay within the key storage budget. Our analysis in Section 5 shows that such a randomized selection scheme can indeed balance effective filtering and strong resiliency, at a moderate storage overhead.

Accommodating node failures The design needs to further handle node failures, which may disrupt an operational forwarding path. Fortunately, geographic routing can typically find detours and bypass the failing nodes via perimeter routing. Under moderate node failures, these detoured paths are still close to each other. This leads us to model the forwarding path with a generic beam in accommodating node failures. In this model, the possible forwarding 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 failure "hole" up to a diameter of b in the forwarding path. We will validate this model and discuss its impact in Section 6.

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 d-distance away from the sink. From geometry we can calculate the spanning angle between l_1 and l_2 as

$$\alpha = 2 \arcsin \frac{b}{max(b, 2d)} \quad (4)$$

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 the sink and itself. In general, a node closer to the sink has a larger upstream region, because reports originating from different locations will converge around the sink.

The choice of b, the beam width, affects the system resiliency and the delivery ratio of legitimate reports. As shown later, the number of keys stored at each node grows linearly with b. This is because with a wider beam, each node has a larger upstream region, thus stores more keys. As a result, the attacker can obtain more keys by compromising a node, a negative impact on the system resiliency. On the other hand, because a forwarding node always drops those reports originated from areas outside its upstream region, a narrow beam may result in unnecessary dropping of legitimate reports when they traverse outside the expected beam, e.g., due to massive node failures. We evaluate the impact of b using analysis and simulations in later sections.

	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 per cell	12 nodes
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

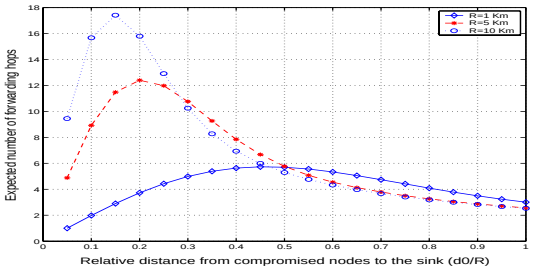

Figure 3: LBRS quickly drops forged reports en-route, and its filtering power scales well to the network size.

5. ANALYSIS

In this section we analyze the performance of our design. We start with the filtering power of LBRS 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. The analysis results quantify the resiliency, efficiency, and scalability of LBRS.

To simplify the analysis, we consider a circular terrain with a radius of R, over which N sensor nodes 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 shapes, such as rectangles, and sink locations as well. However, the presentation will be more involved. Table 1 summarizes the notations used hereinafter, and the default parameter settings used in our numeric evaluations.

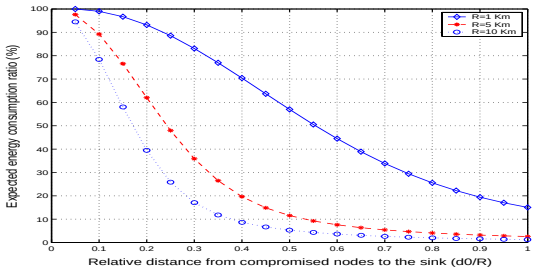

Figure 4: LBRS significantly saves energy by filtering forged reports en-route, especially in large networks.

5.1 Filtering Effectiveness

We analyze the filtering performance of LBRS using two metrics: (1) *detection ratio*: the percentage of forged reports that are detected and dropped, and (2) *filtering position*: the number of hops a forged report can traverse before being dropped.

Consider a base setting where there is a single compromised node (or equivalently, non-colluding compromised nodes). Let node Z be the compromised node, with a distance of d_0 to the sink. A forged 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.

Detection Ratio Because the compromised node has at most one key for any cell, it has to forge at least $m-1$ MACs ($8s(m-1)$ bits), 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.

Filtering Position LBRS can quickly filter the forged 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 forged report can traverse, is upper bounded as:*

$$h' \leq 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} \left(1 - \frac{(m-1)(d_0 - jR_c)}{RL}\right) \qquad (5)$$

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 LBRS, 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 LBRS scales very well when the network size increases.

Energy Saving The early dropping of forged reports leads to significant energy savings in large 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 LBRS-protected paths and the unprotected paths, i.e., h'/h. The figure shows that on average LBRS can save energy by a ratio of 43.7% in a 1Km-radius network, and 81.3% in a 10Km-radius network. The reason for such an increase in the energy saving ratio is that the filtering position in LBRS increases much slower than the network size.

Figure 4 also shows that the energy savings of LBRS depend 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 forged reports, which are dropped more quickly.

5.2 Resiliency in Graceful Degradation

Now we analyze the resiliency of LBRS 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 an arbitrary cell (X, Y). We will show that the security protection offered by LBRS degrades gracefully, rather than completely breaks down in the entire network as in existing designs [23, 28].

Note that 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 successfully 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 from them. This way, the sink may lose monitoring capability in the compromised cells, but the rest of the network is still protected by LBRS.

There are two cases for multiple compromised nodes: they are randomly distributed, or co-located in the same cell. The location-guided key selection ensures that the chance of cell compromise is extremely low when nodes are randomly compromised. In fact, the simulation results in Section 6 show that in such cases, the chance of cell compromise decreases *exponentially* with respect to m. Each additional MAC carried in the reports can reduce the probability of cell compromise by an order of magnitude.

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 LBRS 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 verifiable cell on average. When the attacker forges events in such remote cells, the degradation of LBRS's filtering power is characterized in Theorem 2 (proof in Appendix).

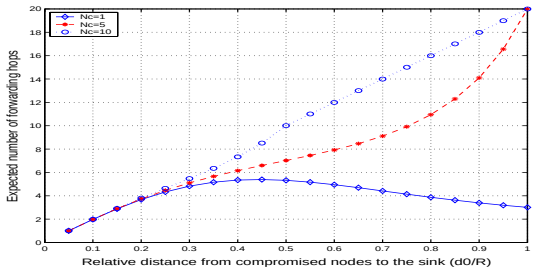

Figure 5: The performance of LBRS degrades gracefully even in the worst-case scenarios.

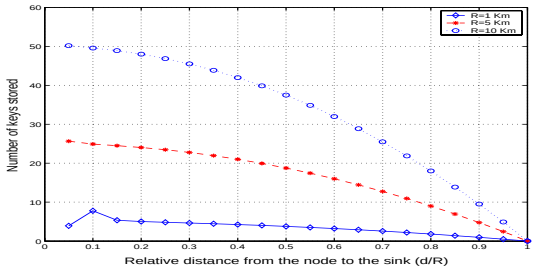

Figure 6: Each node stores only a small number of keys, and the key storage overhead scales well in large networks.

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

$$h' \leq 1 + \sum_{i=2}^{h} \prod_{j=1}^{i-1} (1 - \frac{(mR - d_0 N_c)(d_0 - jR_c)}{R^2 L}) \qquad (6)$$

Figure 5 illustrates the above graceful performance degradation in the worst-case scenarios. In this figure, we fix the node population as 4K and the 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 forged reports increases only slightly as more nodes are compromised. For example, when N_c increases to 5, on average LBRS can still filter forged reports in 7.3 hops, leading to 27% energy savings. We emphasize that this is a worst-case analysis, and LBRS is much more effective in average cases, which we will show in Section 6 using simulations.

5.3 Key Storage Overhead

In LBRS, each node stores one key 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 characterize the key storage overhead in the following theorem (proof in Appendix).

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

$$N_{key} \approx \frac{d(R^2 - d^2)}{2RC^2} \times \arcsin \frac{b}{max(b, 2d)} \approx O(\frac{bR}{C^2}) \qquad (7)$$

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

Despite its strong filtering power, LBRS 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 LBRS. 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 constraint of existing sensor hardware.

5.4 Impact of Other Attacks

LBRS focuses on event fabrication attacks launched by compromised nodes. Our intention is to demonstrate, through LBRS, how to achieve resilient security through the location-based design approach. There are certainly many other attacks that LBRS cannot, and is not designed to, defend against. Nevertheless, we discuss below the impact of two relevant attacks on LBRS, and how we may handle them.

Report Disruption Attack An attacker may launch several attacks to disrupt the legitimate reports. These attacks include 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* [3, 14] in which a compromised node presents multiple identities and announces one incorrect MAC in each identity. As a result, the final report on a real event may be poisoned by incorrect MACs, and dropped in delivery or finally rejected by the sink.

LBRS can localize the damage of such attacks to the cell where the compromised nodes reside. 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 to the event's location. A local authentication mechanism, e.g., pairwise keys [6] and μTESLA [15], or Sybil defense mechanism [14] 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.

Sensor Relocation Attacks The attacker may physically relocate a node from its original location to a new one. When a real event happens nearby the new location, this node may generate an incorrect report using its original location. There are two possible cases: a) The attacker has already compromised the relocated node. Thus the sensor relocation attacks do not incur additional damage to LBRS, because the attacker is already able to control the compromised node to fabricate any report. b) The attacker has not compromised the relocated node. In such cases, sensor relocation attacks can be defeated by local authentication mechanisms, because the relocated node cannot establish trust

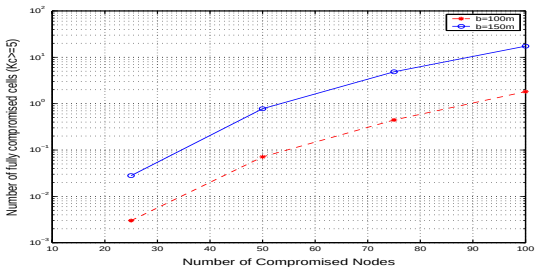

Figure 7: The attacker can hardly collect enough keys bound to a cell when the compromised nodes are randomly scattered in the network.

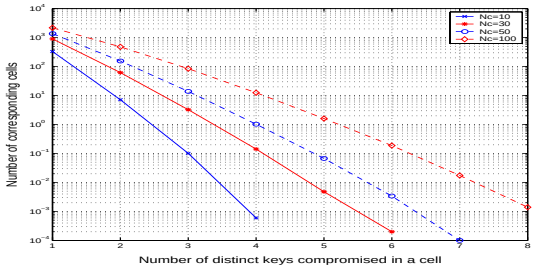

Figure 8: The difficulty to obtain more distinct keys of a cell increases exponentially.

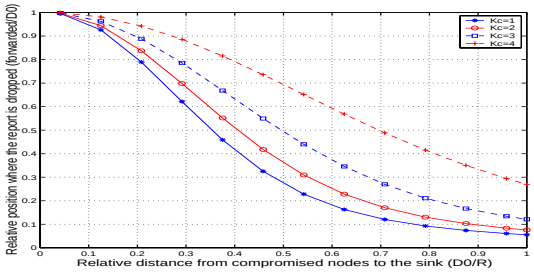

Figure 9: The filtering power of LBRS degrades gracefully when the attacker has collected more keys of a cell.

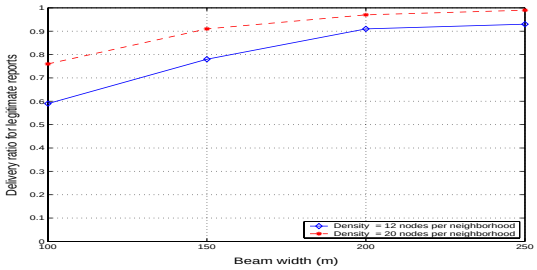

Figure 10: With a moderate beam width, most legitimate reports can be successfully delivered to the sink.

with its new neighbors, hence its reports will be dropped at the first hop.

6. SIMULATION EVALUATION

In this section, we evaluate the performance of LBRS through simulations that complement our analysis. Specifically, we evaluate the resiliency of LBRS under random node compromises, and validate the beam model on geographic forwarding in the presence of node failures.

Resiliency to random node compromise Given that we have analyzed the worst-case resiliency of LBRS when multiple compromised nodes are within the same cell, we are interested to use simulations to study its average-case performance when multiple compromised nodes are randomly distributed. For this purpose, we developed our own simulation platform using Parsec, mainly because other simulators scale poorly to large numbers of nodes. Our simulator implemented the basic geographic forwarding [9] and the LBRS protocol stack. We simulated rectangular terrains to complement our circular terrain based analysis. The parameter settings are similar to those in Table 1, unless explicitly stated. Our simulation results show that LBRS is highly resilient to random node compromise.

We first study how many cells can be compromised, and to what extent, by an attacker combining keys from multiple compromised nodes. In the simulations, 30K nodes are spread over a 5Km× 5Km field, divided into 100m×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.

The simulation results show that even by combining keys from 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 that cell. We can see that with a beam width of 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 reduces the damage to 1.8 fully compromised cells on average.

Figure 8 provides a more detailed view on the aggregated effects of combining keys from multiple compromised nodes. The figure plots, in a log-linear manner, how many cells have a given number of keys disclosed, when $10, 30, 50, 100$ nodes are compromised respectively. The Y axis is the number of cells who has X keys disclosed. We can see that the number of cells with x keys disclosed decreases almost exponentially when x increases. For instance, when 100 nodes are compromised, there are about 12 cells having 4 keys disclosed, but only 1.8 nodes having 5 keys disclosed. Therefore, each additional MAC carried in the report can decrease the chance of cell compromise (hence successful report fabrication) by about an order of magnitude. By requiring each report to carry more MACs (i.e., increasing m), we can significantly enhance the resiliency of LBRS against a large collection of randomly compromised nodes.

Next we simulate how the filtering power of LBRS degrades when the attacker has obtained a few keys of a cell. In the simulations, we vary the locations where the fabricated reports are injected, from adjacent to the sink to network edge. Since each report carries only 5 MACs, there is no need to simulate the cases when the compromised nodes collectively have 5 or more keys of the same cell. The simulation results are shown in Figure 9. We can see that with each

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

additional compromised key, the decrease of filtering power is only marginal, leading to graceful performance degradation.

On the beam model Now we verify the effectiveness of the beam forwarding model in accommodating node failures. 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 as 100m, 150m, 200m, and 250m, and simulate different node failure cases. The effective node density varies 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 97.6% 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. That is, the beam width is roughly four-hop communication range. Clearly the delivery ratio depends on both the beam width and the node density, and 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.

7. PROTOTYPE IMPLEMENTATION

We implemented the LBRS design on MICA2 motes developed by X-Bow. These tiny devices are equipped with an 8-bit 4MHz microcontroller running a microthread operating system, called TinyOS, 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 [17]. This module facilitates the derivation of location-binding keys, as well as the generation and verification of MACs. We also implemented a generic wireless communication module to exceed the packet size limit of 29 bytes in the TinyOS *GenericComm* interface. It directly reads and writes the buffer associated with the low-level radio device, and can transmit packets of any length.

Code size Table 2 shows a breakdown of the implementation code size on the MICA2 platform. The LBRS 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 and cryptography modules, timer and sensing 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.

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 afterward, posing high time constraints for an attacker to compromise the master key. The MAC generation and verification are also fast: 10 ms to generate or verify a MAC for 24-byte data reports.

8. DISCUSSION

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

Sensor deployment Sensor nodes can be deployed in different ways. In many cases, their deployed locations are not known a priori, e.g., when they are dropped via vehicles or aircrafts. In such cases, LBRS needs a secure localization protocol, so that sensors can securely obtain their locations with certain accuracy. In fact, secure localization is required in all applications that work in an hostile environment to tag events with correct locations. A number of proposals [10, 13, 11] have started to address this problem.

In LBRS, the master secrets must be protected during the bootstrapping phase of the new nodes. Because the bootstrapping phase is typically very short (e.g., a few seconds), the chance for successful attacks is very limited. Appropriate network planning and deployment can also keep away attackers during the bootstrapping process. We can further protect the master secrets by setting a timer at each newly deployed node, which erases the master secret upon timeouts even though it has not been fully bootstrapped. This may lead some nodes to be useless; however, given the high density, the network can still function well as a whole.

Sensor nodes can also be deployed to pre-determined locations, where deployment is carefully planned or security requirement is stringent, for instance, for in-building or highway monitoring applications. With such a deployment model, we can preload the sensors directly with location-binding keys, rather than the master secret, because the deployed locations of the nodes are known *a priori*. As a result, the secret keys can be strictly protected.

Node density Topology control protocols [24] are commonly used to prolong the sensor network lifetime by turning redundant nodes into sleeping. To allow enough sensing nodes to jointly generate a report, a sleeping node can leave its sensing module on and turn off the communication module, the dominant energy consumer. Once an event happens, nearby nodes wake up, triggered by the sensing module, and collaborate in generating the reports. This way, LBRS can still achieve energy efficiency and resilient security.

Routing The upstream region estimation in LBRS is designed to work with geographic routing protocols. We conducted experiments and found several non-geographic sensor routing protocols, such as Directed Diffusion [8] and GRAB [21], also fit well with the beam forwarding model. They can potentially work with LBRS. However, a thorough investigation is needed and we leave it for future research.

In concave terrains (e.g., with half-moon shapes), the beam forwarding model may not work well as the forwarding paths cannot be accurately approximated as straight lines. In such cases, a node can use different strategies, such as a uniform one, in selecting the verifiable cells. The tradeoff between resiliency and data delivery needs further investigation.

Key update and revocation Currently LBRS does not provide any key update or revocation mechanism. Re-

cent work [25] has started to address sensor re-keying. We plan to extend LBRS by binding keys to a spatial-temporal space, i.e., a combination of geographic location and time, through mechanisms such as hash chains [15]. Thus we can update or revoke the keys either periodically or upon security compromises.

9. RELATED WORK

Security is essential for sensor networks to work in practice, in particular over adverse or hostile environments. There have been many proposals studying various aspects of sensor network security. We briefly summarize and compare the most related ones with LBRS.

Key management is among the first topics explored in sensor network security. A number of pairwise key establishment schemes [6, 2, 5, 4, 12, 27] have been proposed. They provide basic authentication, confidentiality and prevent outsiders from attacking the network. They use the idea of probabilistic key sharing [6] to establish trust between two nodes, with different emphasis on enhanced security protection [2], flexibility of security requirements [27], high probability of key establishment and reduced overhead [12], or utilization of deployment knowledge [4]. However, they are not designed to handle insider attacks, such as event fabrication, launched by compromised nodes. A compromised node already possesses correct keys to authenticate its message, and it can fabricate events arbitrarily. Other sensors and the sink cannot distinguish forged reports from real ones. LBRS differ from all these solutions in its capability to deal with insider attacks.

Two recent proposals SEF [23] and IHA [28] provide limited protection against insider attacks through probabilistic key sharing over a partitioned key pool and interleaved per-hop authentication, respectively. However, both solutions are not resilient in that they completely lose the security protection when the attacker has compromised more than a small, fixed number of nodes. LBRS eliminates such *threshold breakdown* by exploiting a location-based approach as the fundamental mechanism towards resilient security. To our best knowledge, LBRS is the first security solution that can achieve graceful performance degradation to an increasing number of compromised nodes.

The compromised nodes may launch other insider attacks than event fabrication attacks. For example, they can attack the commonly used in-network aggregation mechanism by producing false aggregation results. A secure aggregation mechanism is proposed in SIA [16]. However, this problem is different from event fabrication attacks in which the compromised nodes forge reports, i.e., raw data, in the first place.

10. CONCLUSION

Node compromise presents severe security threats in sensor networks. Existing solutions either do not address such insider attacks, or completely break down when more than a fixed threshold number of nodes are compromised. LBRS aims at providing resilient security and graceful performance degradation against an increasing number of compromised nodes. It achieves resiliency by limiting the scope for which keys are used. Different from the existing work that binds keys to nodes, LBRS binds keys to geographical locations. This ensures that the keys can only be used to endorse local events where they are bound. The attacker can no longer abuse the compromised keys for global usage, such as fabricating events in arbitrary locations.

As one general design guideline, constraining the scope for which secrets are used can lead to higher degree of resiliency. However, in symmetric-key based designs, the same secret key is used for two different functions: credential generation and verification. Had these two functions relied on different secrets (e.g., as in public-key cryptography), compromise of verifying nodes leads to little harm because the verification secret cannot be used to forge credentials. Our location-binding keys offer an alternative way to limit the scope of key usage. We have demonstrated, through LBRS, that such a location-based design approach can achieve resilient security in an efficient and scalable fashion. It provides a 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.

11. ACKNOWLEDGMENTS

The authors are indebted to Jerry Cheng for his help on the simulator, and Haiyun Luo for his insightful discussions in the early stage of this work. We would also like to thank the anonymous reviewers and our group members of Wireless Networking Group (WiNG) at UCLA for their constructive criticisms.

12. REFERENCES

[1] R. Anderson, H. Chan, and A. Perrig. Key Infection: Smart Trust for Smart Dust. In *Proc. IEEE International Conference on Network Protocols (ICNP)*, 2004.

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

[3] J. Douceur. The Sybil Attack. In *Proc. International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002.

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

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

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

[7] Q. Fang, J. Gao, and L. Guibas. Locating and Bypassing Routing Holes in Sensor Networks. In *Proc. IEEE INFOCOM*, 2004.

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

[9] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. ACM MOBICOM*, 2000.

[10] L. Lazos and R. Poovendran. SeRLoc: Secure Range-Independent Localization for Wireless Sensor Networks. In *Proc. ACM Workshop on Wireless Security (WiSe)*, 2004.

[11] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust Statistical Methods for Securing Wireless Localization in Sensor Networks. In *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

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

[13] D. Liu, P. Ning, and W. Du. Attack-Resistant Location Estimation in Sensor Networks. In *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[14] J. Newsome, R. Shi, D. Song, and A. Perrig. The Sybil Attack in Sensor Networks: Analysis and Defenses. In *Proc. International Symposium on Information Processing in Sensor Networks (IPSN)*, 2004.

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

[16] B. Pryzdatek, D. Song, and A. Perrig. SIA: Secure Information Aggregation in Sensor Networks. In *Proc. ACM SenSyS*, 2003.

[17] R. Rivest. The RC5 Encryption Algorithm. In *Workshop on Fast Software Encryption*, 1995.

[18] Y. Shang, W. Ruml, and Y. Zhang. Localization from Mere Connectivity. In *Proc. ACM MOBIHOC*, 2003.

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

[20] A. Wood and J. Stankovic. Denial of Service in Sensor Networks. *IEEE Computer*, October 2002.

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

[22] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks. In *Proc. ACM MOIBCOM*, 2002.

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

[24] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS:A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proc. IEEE ICDCS*, 2003.

[25] W. Zhang and G. Cao. Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration Based Approach. In *Proc. IEEE INFOCOM*, 2005.

[26] 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.

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

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

13. 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.

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

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 4.3 (See Figure 2 for a graphical illustration). Recall that we have derived the spanning angle of Γ, denoted by α, in Equation 4.

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,

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