Least Privilege and Privilege Deprivation: Towards Tolerating Mobile Sink Compromises in Wireless Sensor Networks

Wensheng Zhang, Hui Song, Sencun Zhu*, and Guohong Cao
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
Email: {wezhang, hsong, szhu, gcao}@cse.psu.edu

ABSTRACT

Mobile sinks are needed in many sensor network applications for efficient data collection, data querying, localized sensor reprogramming, identifying and revoking compromised sensors, and other network maintenance. Employing mobile sinks however raises a new security challenge: if a mobile sink is given too many privileges, it will become very attractive for attack and compromise. Using a compromised mobile sink, an adversary may easily bring down or even take over the sensor network. Thus, security mechanisms that can tolerate mobile sink compromises are essential. In this paper, based on the principle of least privilege, we first propose several efficient schemes to restrict the privilege of a mobile sink without impeding its capability of carrying out any authorized operations for an assigned task. To further reduce the possible damages caused by a compromised mobile sink, we then propose efficient message forwarding schemes for depriving the privilege assigned to a compromised mobile sink immediately after its compromise has been detected. Through detailed analysis and simulations, we show that our schemes are secure and efficient, and are highly practical for sensor networks consisting of the current generation of sensors.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Miscellaneous

General Terms

Security, Algorithm

Keywords

Wireless Sensor Networks, Mobile Sinks, Privilege

*also with School of Information Sciences and Technology, The Pennsylvania State University

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

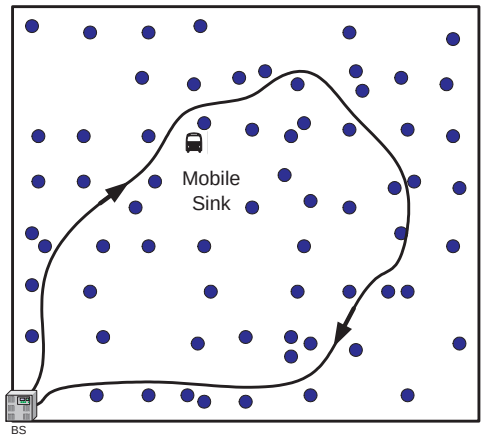

Figure 1: An example application where a MS is dispatched to carry out a task along a predetermined trajectory

1. INTRODUCTION

Mobile sinks (or mobile solders, mobile sensors, shown in Figure 1) are one of the essential components for the operation of many sensor network applications. One such application is data collection. Wireless sensor networks [1] allow continuous environment monitoring in hazard or remote areas. The sensed data often need to be sent back to the base station for analyzing. However, when the sensing field is too far away from the base station, transmitting the data over long distance to the station may increase the delay, add more network traffic, and weaken the security strength (e.g., some intermediate may modify the data passing by). To address these problems, researchers [2, 3] have proposed to temporarily store the data in sensor nodes, and let the base station periodically dispatch mobile sinks (MSs) to collect data. In some strategic scenarios such as battlefield, MSs [4, 5] are even allowed to directly query data from sensor nodes at any time. In addition to be useful for data collection, MSs may also be employed for network management. For example, the base station may send MSs to detect nodes being compromised or to repair failed nodes.

The use of a MS however introduces a new security challenge: the privilege granted to a MS can be abused once it is compromised. This will become a serious problem if a MS is granted many privileges, making it very attractive for attack and compromise. For example, suppose we want to send a MS to collect sensor readings in a specific location during a specific time interval. Without ap-

propriate restrictions, a compromised MS may be able to collect sensor data from the entire network at all time. In another example, suppose we want to dispatch a MS to inspect an abnormal area of a sensor network and the MS is allowed to revoke and isolate a sensor node if the sensor node is identified as compromised. Again, without appropriate restrictions, a compromised MS can easily revoke any nodes of its choice and bring down the entire network by simply sending some revocation messages. The severe consequence of MS compromises can also be foreseen in other applications, which indicates the importance of restricting the privilege of MSs. On the other hand, the convenience of a MS in executing the authorized operations should not be scarified by the increased security restriction.

We can limit the privilege of a MS through policy, but the enforcement of the policy cannot rely on the trustworthiness of the MS because the MS may be compromised. Therefore, we have to resort to security mechanisms for policy enforcement. The design challenge arises from the fact that sensor nodes are deployed prior to the dispatch of MSs. For some applications, sensor nodes do not know in advance the policy for an individual MS due to the large variety and the on-demand nature of the tasks. One solution could be flooding the entire network to notify all the sensor nodes of the task as well as the privilege of the MS to be dispatched. However, this not only increases the communication overhead, but also requires every sensor node to receive and store information regarding the MS. Moreover, The overhead increases linearly with the number of MSs employed in the system.

To the best of our knowledge, the issue of tolerating MS compromise has not been addressed in the literature. Previous research on sensor network security has been focused on broadcast source authentication [6, 7], key management [8, 9, 10, 11, 12], and others [13, 14, 15]. These schemes may be employed for securing the communication between regular sensor nodes or between a MS and a regular sensor, but they cannot restrict the privilege of a MS while giving enough privilege for the MS to accomplish its assigned task.

Contributions We propose two sets of solutions to address the MS compromise problem. First, based on the principle of least privilege [16], we design several security restriction schemes which only grants the MSs the least privilege required to accomplish their tasks. These schemes *allow* and *only allow* a MS to perform the pre-authorized operations. A salient feature of our schemes is that neither do we need to pre-load the sensor nodes with the tasks that might be carried out by MSs, nor do we need to flood the network when dispatching a MS. Indeed, our constructions guarantee that i) a MS cannot lie or modify its assigned task, and ii) without any knowledge about the task any sensor node can verify if a claimed task is valid. If the verification fails, a sensor node rejects the request from the MS. We show through detailed analysis that these schemes are very effective and efficient.

Second, we propose several on-demand schemes for privilege deprivation: a basic scheme followed by three optimized schemes which can revoke the privilege granted to the MS. Privilege deprivation is important and necessary when a MS has been compromised or the security policy has been changed. Through detailed simulation study, we show that our optimized schemes can greatly reduce the revocation latency as well as the communication overhead compared to the basic scheme.

Organization We describe our assumptions on node, network, and security as well as our design goal in Section 2. Section 3 presents our schemes, four security restriction schemes in Section 3.1 and four privilege revocation schemes in Section 3.2. Section 4 describes some related work. Finally, Section 5 concludes the paper and discusses several future research directions.

2. NETWORK AND SECURITY ASSUMPTIONS

2.1 Node and Network Assumptions

We assume regular stationary sensor nodes are constrained in resources. Our proposed schemes target at the current generation of sensor nodes such as MICA/MICA2 motes developed at UC Berkeley [17]. The mote runs the special operating system called TinyOS, which supports the default packet size of 36 bytes out of which 29 bytes are for the actual payload. We assume that every node has space to store several hundred bytes of keying information. A mobile sink (MS) can be as powerful as a laptop-class device or a PDA; however, to make our scheme more general, we also consider resource-constrained mobile devices such as mobile sensors [18] that have the same amount of resources as the Mica2 Motes. A base station (BS) is located in a fixed and secure position. It is more powerful than MSs and cannot be compromised.

We consider a sensor network divided into grids or cells. Each cell has a unique id and every sensor node knows in which cell it is located [19]. We consider the type of applications in which we dispatch a MS with a known purpose. More specifically, a MS is assumed to perform some tasks along a roughly predetermined physical path (may be an arbitrary curve) in a sensor network. We know the type of a task (e.g., collecting data or network diagnose), the start time and the end time, and the cells involved in the task.

We assume the clocks of sensor nodes in a network are loosely synchronized. Time synchronization is important for general sensor network applications such as mobile object tracking, data aggregation, TDMA radio scheduling, message ordering. It is also needed in our scheme since a sensor node needs to verify if the task claimed by a MS has the valid time interval. Existing time synchronization protocols [20] are shown to be able to provide millisecond-level precision.

2.2 Security Assumptions

We assume that the base station has a mechanism to authenticate broadcast messages (e.g., based on μTESLA [7]), and every node can verify the broadcast messages. We also assume that when an adversary compromises a node, either a regular node or a MS, it can obtain all the sensitive keying materials possessed by the compromised node. Moreover, the adversary may pool the keying materials from multiple compromised nodes to break the security of the network or to launch advanced attacks. However, we assume that the base station will not be compromised.

Since wireless communication is broadcast-based, we assume that an adversary can eavesdrop on all traffic, inject packets, and replay older packets. We note that an adversary may try to attack the node localization protocol and the time synchronization protocol employed in the sensor network to gain advantages, as a task is location- and time-specific in our proposed schemes. The design of attack-resilient location protocols and time synchronization protocol is out of the scope of our work. Attacks and countermeasures for range-dependent and range-independent node localization schemes have been presented in [21] and in [22], respectively.

The motivation of privilege deprivation (or interchangeably called MS revocation) can be due to the change of security policy, the detection of the MS compromises, or other reasons. In particular, we do not assume that the reason for a MS revocation must be the detection of its misbehavior (e.g, injecting spurious packets). In some applications, MSs are devices carried by other entities (e.g., soldiers, vehicles). Therefore, a MS revocation is often the result of revoking the carrier of the MS. For example, if a mobile solider is captured by the adversary or is missing in a battlefield, other sol-

diers can report the event to the network controller, which initiates a MS revocation operation to revoke the MSs carried by this solider.

2.3 Design Goal

Our goal is to design security mechanisms to minimize the potential damages caused by a compromised MS, thus tolerating MS compromises. More specifically, we will propose security mechanisms to meet the following requirements.

- **Least Privilege** A MS should be granted the least privilege required to accomplish its task. Specifically, we should load the MS with the minimal number of keys or security credentials that allow it to communicate with sensor nodes securely.
- **Immediate Privilege Deprivation** Once the compromise of a MS is detected, it should be revoked immediately rather than waiting for the authorized time period to expire.
- **On-demand** We should be able to assign a task on demand as long as the type, the locations and the time interval of a task are valid. The sensor nodes in a sensor network do not need to know a task before deployment or be notified by a network controller on the fly.
- **Efficiency** Since we are targeting at the resource constrained sensor nodes and MSs, the schemes should be efficient in terms of communication and computation.

2.4 Notations

The following notations appear in the rest of this discussion.

- u (in lower case) is a regular stationary sensor node.
- MS is a mobile sink.
- TT is the type of a task.
- T_s, T_e are the starting time and the ending time of a task, respectively.
- $MAC(k, s)$ is the message authentication code (MAC) of message s computed with a symmetric key k.

In addition, a sensor node is referred to as a *host node* of a MS if the MS is authorized to talk to it in a task.

3. PROPOSED SCHEMES

To restrict the privilege of a MS, we must enable sensor nodes to validate the task claimed by the MS. If the validation fails, sensor nodes will reject any requests from the MS. Thus a MS can only carry out its authorized task. As a MS is attractive for attack and compromise, in addition to adopting a prevention-based privilege restriction approach, we also need a proactive revocation approach to prevent the compromised MS from causing further damages to the host nodes.

Next we propose two sets of solutions to address the MS compromise problem. Section 3.1 discusses three schemes (a basic scheme followed by two security enhanced schemes) for restricting the privilege of a MS; Section 3.2 presents four message forwarding schemes (a basic scheme followed by three optimization schemes) for efficiently and quickly deliver a MS revocation notification to all the host nodes of the compromised MS.

3.1 Restricting the Privilege of a MS

3.1.1 Scheme I: The Basic Scheme

In the basic scheme, every node is pre-loaded with an individual key shared with the base station (BS). BS generates a master key K_m, based on which it derives an individual key for every node u as $K_u = G_{K_m}(u)$, where G is a pseudo-random function (PRF) [23]. Now suppose BS knows in advance the ids of all the host nodes for the MS in the task. BS loads the MS with a pairwise key $K_u(MS)$ shared with each host node u.

$$K_u(MS) = H(TT|MS|K_u|T_s|T_e), \tag{1}$$

where H is a collision-resistant one-way hash function and $|$ denotes the concatenation of messages.

To establish a pairwise key with node u, the MS needs to send node u its id MS, TT, T_s, and T_e, based on which node u can compute $K_u(MS)$ in the same way. Next node u and MS authenticate to each other by, for example, exchanging the following authentication messages

$$MS \rightarrow u : R_1, MAC(K_u(MS), R_1) \tag{2}$$

$$u \rightarrow MS : R_2, MAC(K_u(MS), R_1|R_2). \tag{3}$$

Here R_1 and R_2 are randomly generated nonces for preventing replay attacks. If node u can successfully verify the message from MS, it will trust the MS and hence assist the MS in the task of type TT during the time interval $[T_s, T_e]$; otherwise, it knows the claimed task is not authorized. Note that due to the one-way property of function H, a compromised MS cannot forge any of the values in (TT, MS, T_s, T_e).

The basic scheme meets our design goal and should work well if a MS only communicates with a small number of host nodes. The scheme however is not scalable in terms of storage if the MS is expected to talk to a large number of host nodes, because the MS needs to store one pairwise key shared with each host node. A more concerned limitation is the requirement of global knowledge on the network topology. Although the BS approximately knows the locations that the MS should visit and it can design an appropriate trajectory for the MS, it might not know the ids of the host nodes (i.e., nodes located on the trajectory). Thus, the BS cannot pre-load the MS with the pairwise keys shared with the en-route host nodes.

3.1.2 Scheme II: A Blundo Scheme-based Construction

The limitations in Scheme I are due to the lack of scalability of the PRF-based pairwise key pre-deployment scheme. A scalable scheme for establishing pairwise keys should have the following properties. First, the number of preloaded keys should not increase with the number of host nodes. Second, the scheme should not rely on the pre-knowledge of the ids of the host nodes. Thus we need an on-demand scheme which allows a MS to establish a pairwise key with any sensor node on-the-fly.

Recently many pairwise key establishing schemes [8, 9, 10, 11, 12] have been proposed. All these schemes enable two nodes to establish a pairwise key on the fly once the two nodes know each other's id, although they provide different security guarantees and incur different performance overheads. In this work, we choose the Blundo scheme [24] to construct our protocols, although several other schemes [9, 11] might as well be adapted for our purpose. As we shall see shortly that the Blundo scheme provides clear security guarantee. Therefore, the use of the Blundo scheme greatly eases the presentation of our work and enables us to provide a clearer security analysis.

The Blundo scheme, when applied to ad hoc or sensor networks, usually involves the following steps.

- The BS (or a key server) chooses a random symmetric bivariate polynomial $f(x, y)$ of degree t with coefficients over a finite field $GF(q)$, where q is a prime number large enough to accommodate a symmetric key.

$$f(x,y) = \sum_{0 \le i,j \le t} a_{ij} x^i y^j, \tag{4}$$

  where $a_{ij} = a_{ji}$.

- The BS loads every node u with $f(u, y)$, which is a polynomial obtained by evaluating $f(x, y)$ at $x = u$.

$$f(u,y) = \sum_{0 \le i \le t} a_i(u) y^i. \tag{5}$$

- If two nodes u and v want to set up a pairwise key, each of them evaluates the other's id in its own polynomial. The result $f(u, v) = f(v, u)$ serves as their pairwise key.

Suppose every node in the network has been loaded with its polynomial share before its deployment. Every node can establish a pairwise key with every neighbor node or any other node in the network if needed. After the BS gives a share of the polynomial $f(x, y)$ to the MS (i.e., $f(MS, y)$), the MS can establish pairwise keys with any node in the network on the fly without knowing the id of that node in advance, thus addressing the limitation in Scheme I.

The security of the Blundo scheme is determined by t. The scheme provides unconditional secrecy if no more than t nodes collude. If more than t nodes collude by pooling their shares, they can recover the polynomial $f(x, y)$ and hence break the system. Therefore, t should be large enough for the application under consideration. On the other hand, a node stores a univariate polynomial share represented by $t + 1$ coefficients. The size of a coefficient is the same as that of a symmetric key. For the current generation of sensor nodes with $4K$ RAM, t could be up to several hundreds under the memory constraint [6, 9].

This scheme if applied directly has one drawback. It does not limit the privilege of the MS node. The MS can establish a pairwise key with any node in the network, thus its compromise will lead to a global disaster. To prevent MS from establishing pairwise keys with arbitrarily selected nodes in the network, we propose to embed some information about the host nodes into the id of the MS so that a sensor node can verify if it is a host node for the MS. An example construction is as follows. Suppose node u is a host node for MS. The BS constructs the id of the MS as

$$MS(u) = H(TT|T_s|T_e|u). \tag{6}$$

The BS then pre-loads MS with a polynomial share $f(MS(u), y)$. To establish a pairwise key with node u, the MS sends (TT, Ts, Te) to u. Node u can then derive $MS(u)$ in the same way. Next both MS and node u compute their pairwise key $f(MS(u), u)$ (=$f(u, MS(u))$). Finally, they can authenticate each other by exchanging authentication messages as shown in Scheme I.

The scheme however still has several limitations. First, storage is still a concern. If the MS is going to communicate with m nodes, it needs to store $m(t + 1)$ coefficients. Here t could be large because of the security consideration in the Blundo scheme. For example, if $t = 50$, $m = 100$, and the size of a coefficient is 8 bytes, the MS needs to store about 40 KB keying material. This precludes the use of low-end mobile sensors as MSs. Second, more importantly, loading MS with multiple polynomials greatly endangers the polynomial $f(x, y)$ because an attacker can get multiple shares of the polynomial once it compromises the MS. As an extreme example, if m is larger than t, an attacker will be able to reconstruct $f(x, y)$ by solely compromising the MS.

3.1.3 *Scheme III: Reducing The Number of Polynomial Shares To One*

To address the remaining issues of Scheme II, we reduce the number of polynomial shares possessed by a MS — ideally a MS only possesses *one* polynomial share. In this section we present a construction that achieves this goal. There are two techniques. First, we use the locations of host nodes, instead of their ids, to reduce the information of the host nodes that has to be stored by a MS. Second, we use a Merkle hash tree [25] to construct the id for a MS, so that only one polynomial share has to be assigned to the MS. Below we describe these two techniques in more detail.

Cell Merging A network field is divided into cells and a cell is referenced by an index (i, j). BS is located at cell $(0, 0)$, as shown in Figure 2. If the MS is scheduled to cross cell (i, j), BS can generate a specific id for MS, $MS(i, j)$.

$$MS(i,j) = H(TT|T_s|T_e|i|j) \tag{7}$$

Now the MS can establish a pairwise key $f(MS(i, j), u)$ with any node u in the cell (i, j) in a way similar to that in Scheme II. Due to the one-way property of function H, MS cannot forge any one of the values in (TT, T_s, T_e, i, j).

Using cells instead of host node ids can reduce the number of polynomial shares possessed by a MS if every cell on average includes multiple sensor nodes. The number, however, might still be large because the size of a cell cannot be too large. In an extreme case, if there is only one cell for the entire network, the privilege of the MS will not be restricted because it can establish a pairwise key with any node in the network.

To further reduce the number of polynomial shares for a MS, we propose an encoding algorithm to merge contiguous cells into blocks. We denote the id of each block as a four-variable tuple (i, j, d, s), where (i, j) is the index of the left-bottom cell (called base cell) in a block, d is the locations of other cells relative to the base cell, and s is the number of other cells in the direction d. d has only two values, '0' denoting top and '1' denoting right; therefore, it can be represented by one bit. $s = 0$ means that a block only has one cell, the base cell. In Figure 2, the MS traverses 139 cells. Using the block representation, these 139 cells can be merged into 57 blocks. For example, the first block is $(0, 0, 0, 5)$ and the second one along the trajectory is $(1, 5, 1, 3)$.

The algorithm runs in multiple iterations. Starting from the base station cell (0,0), we process one row and one column in each iteration. Specifically, the i^{th} row and the i^{th} column is processed in the i^{th} iteration. We find the first horizontal block in the i^{th} row and the first vertical block in the i^{th} column. If the vertical block is larger than the horizontal one, the column is processed first; otherwise, the order is reversed. When processing a column, all the vertical blocks in the column are identified. If an identified vertical block has only one cell, we will replace it with a horizontal block starting from this cell. A row is processed in the similar way. The process continues until all blocks have been identified. The cell merging algorithm is formally presented in Figure 3. Figure 2 shows the blocks identified using this algorithm.

Block Compression Our second technique generates a single id for the MS based on a Merkle hash tree [25]. Suppose we have obtained m blocks after running the above cell-merging algorithm. Let the id of block i be B_i. A Merkle hash tree is constructed

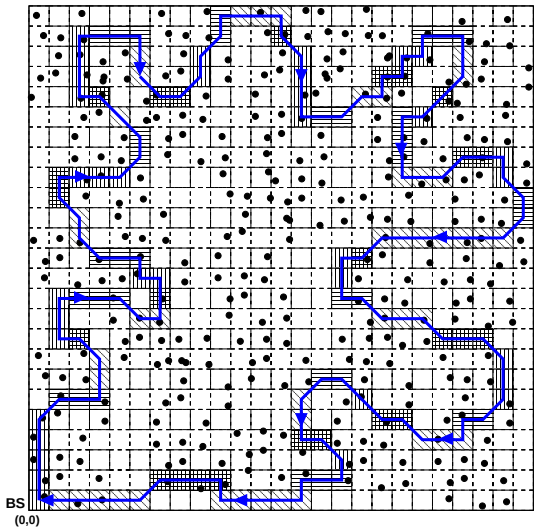

Figure 2: A sensor network field is divided into cells and a mobile sink traverses the field along a predetermined trajectory

in a bottom-up fashion using block ids as the leaf nodes. A non-leaf node in the tree is a hash of its two child nodes, recursively until the root node X_{1m} is generated. Figure 4 depicts an example where $m = 8$. Here $X_{18} = F(X_{14}|X_{58})$, $X_{14} = F(X_{12}|X_{34})$, $X_{34} = F(B_3|B_4)$, and F is a collision-resistant hash function. we can derive the id of the MS node as

$$MS = H(TT|Ts|Te|X_{1m}) \quad (8)$$

To establish a pairwise key with a node u in block B_i, the MS provides MS, Ts, Te, X_{1m} as well as several auxiliary values in the Merkle hash tree allowing node u to verify X_{1m} efficiently. The auxiliary values are the nodes sibling to the nodes on the path from B_i to the root X_{im}. Suppose node u is in block B_3 in Figure 4. MS provides node u with B_3, B_4, X_{12}, and X_{58}. Node u first checks if its own cell is in block B_3. If so, it derives $X_{18} = F(F(X_{12}|F(B_3|B_4))|X_{58})$; otherwise, it terminates the verification process. Next node u derives the id MS based on Formulae (8) and further computes its pairwise key $f(u, MS)$ shared with the MS. The MS also computes $f(MS, u)$. Finally, as in Scheme I, node u and MS provide mutual authentication using their pairwise keys as the MAC keys.

To establish a pairwise key with another node v in Block B_6, the MS also provides with MS, Ts, Te, X_{1m}, but the auxiliary values it presents are B_5, B_6, X_{78}, X_{14}. Thus node u derives $X_{18} = F(F(X_{14}|(F(B_5|B_6)|X_{78}))$. The remaining steps are the same as in pairwise key establishment with node u. More generally, with the same node id, the MS can establish a pairwise key with any node in these predetermined blocks within a specific time interval. Note that only one share is assigned to the MS and no knowledge about the network topology is required in advance.

3.1.4 *Defending Against Denial-of-Service Attacks*

Scheme III is subject to denial-of-service(DOS) attacks due to very small packet size (29-byte payload in TinyOS [17]) used in sensor networks and mutual authentication being the last step. Recall that the first message sent from a MS to a host node includes the id MS and $log(m)$ auxiliary values besides the others. To be computationally secure, the size of the id MS or an auxiliary value should be at least the same as that of a symmetric key, which is normally 8 bytes for sensor networks. This limits the number of auxiliary values carried in a packet to at most three. As such, the MS has to send the message via multiple packets. A host node has to receive all the packets to reconstruct the message, computes MS, and finally proceeds to the mutual authentication process. Only after the mutual authentication has been done can a host node know if the MS is authenticated or not. That is, a host node cannot verify a received packet immediately. An attacker may exploit this security vulnerability to launch DOS attack against a host node. The attacker can send a large number of false packets to a host node to overrun its buffer and to entangle it in processing false packets and sending authentication messages.

Next we propose two enhancements to Scheme III to address the above security weaknesses. First, a host node u and a MS execute the mutual authentication process before the MS provides the parameters regarding the task.

$$MS \rightarrow u : MS, R_1, MAC(f(MS, u), R_1|MS) \quad (9)$$

$$u \rightarrow MS : R_2, MAC(f(u, MS), R_1|R_2). \quad (10)$$

Notations

- n,(i, j): the network field is divided into $n \times n$ cells, each is represented as a 2-tuple (i, j). In this algorithm, we only consider the cells which the MS can talk to.
- $\mathcal{B}$: the set of blocks generated by this algorithm.
- $b(i, j, d, l)$: the largest vertical (if $d = 0$) or horizontal (if $d = 1$) block in which cell (i, j) is the left-bottom cell and the block contains l ungrouped cells, where a cell is *ungroup* if it is not in any block in $\mathcal{B}$.

The Algorithm

$i = 0$; $j = 0$; $B = \emptyset$
while ($i < n$ and $j < n$)
  find $b(i, j, 0, l_0)$ and $b(i, j, 1, l_1)$
  if ($l_0 \leq l_1$)
    process_row(i,j); process_column(i,j)
  else
    process_column(i,j); process_row(i,j)

process_row(i,j)
  add $b(i, j, 1, l_1)$ to $\mathcal{B}$
  while (existing ungrouped cells on this row)
    find the next horizontal block $b(k, j, 1, l)$
    if ($l > 1$) add $b(k, j, 1, l)$ to $\mathcal{B}$
    else add $b(k, j, 0, l')$ to $\mathcal{B}$
  $i = i + 1$

process_column(i,j)
  add $b(i, j, 0, l_0)$ to $\mathcal{B}$
  while (existing ungrouped cells on this column)
    find the next vertical block $b(i, k, 0, l)$
    if ($l > 1$) add $b(i, k, 0, l)$ to $\mathcal{B}$
    else add $b(i, k, 1, l')$ to $\mathcal{B}$
  $j = j + 1$

Figure 3: The basic algorithm for finding blocks

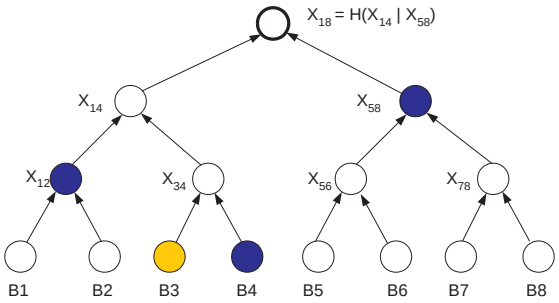

Figure 4: A Merkle-hash tree constructed from the ids of the blocks to be traversed by a mobile sink

When node u receives the message from the MS, it computes its pairwise key shared with MS, then verifies if the message is authenticated. This authentication-first strategy prevents DOS attacks launched by an outsider attacker which does not have a valid polynomial share. However, this scheme still cannot prevent insider attacks because node u cannot tell if the MS is a mobile sink or a compromised regular sensor node in the network.

Our idea to address this problem is to construct the ids for regular sensor nodes and the ids for MS nodes differently so that a host node can tell immediately if the one it is talking to is a MS or a regular sensor node. Recall that the id for a MS is a pseudo-random number output from a hash function and the size of the id is, for example, 8 bytes. Differently, we can choose the ids for regular sensor nodes with certain patterns. A simple pattern is that the ids are integers between 1 and N, where N is the number of sensor nodes in the network. Consider a network size of $N = 65,536$, where a node id can be represented by 2 bytes. Normally the id of a MS will not fall into the interval $[1, 65536]$ because it is unlikely that all the other 6 bytes of a MS id output from a hash function are all zeros. Nevertheless, in case that the rare situation happens, we can change either the value T_s or T_e slightly to derive a MS id falling outside of the interval $[1, N]$ while without sacrificing much security. Thus, this approach can prevent a compromised sensor node from impersonating a MS.

Once a host node is certain that the MS id is valid via the authentication process, it proceeds to verify if it is a host node for the MS and if the MS is authorized for the task to be carried out. A MS is required to provide TT, T_s, T_e, X_{1m} and the auxiliary values, which usually have to be sent in multiple packets. To enable a host node to verify every received packet immediately, for every packet to be sent, the MS provides a MAC using its pairwise key shared with the host node as the MAC key. This prevents any other nodes from injecting packets while impersonating a MS. After obtaining all the packets and deriving X_{1m}, a host node verifies if all the parameters (e.g., time and location) regarding the claimed task is authenticated. If the verification succeeds, it assists the MS for the task; otherwise, it knows the MS is compromised and drops future packets from the MS.

3.1.5 Security and Performance Analysis

Next we analyze the security and the performance of our final scheme (i.e., Scheme III).

Security Analysis The security of Scheme III is based on the assumption that Merkle hash tree, hash function, and the MAC algorithm are secure. In practice, as long as we pick a proper hash function and a MAC algorithm (e.g., using RC5 [26]) and the size of the hash/MAC output is large enough, the scheme guarantees that a compromised MS cannot lie about the type of task, the locations and the time interval in which it is authorized to carry out the task. Since a MS only possesses one polynomial share, an attacker will not gain more advantages from compromising a MS instead of a regular sensor node if its goal is to recover $f(x, y)$. Moreover, Scheme III with the DOS-resistance enhancement enables a regular sensor node to verify every received packet from a MS immediately, thus protecting its packet buffer space from being overflowed by false packets injected by other nodes.

Performance Analysis

- *Computational Cost* Given the auxiliary values in a Merkle hash tree, a host node computes $log(m)$ hashes to get the root value, where m is the number of blocks; it performs one hash computation to derive the id of the MS and two MAC computations (one for generating a MAC and the other for verifying a MAC) during mutual authentication. In practice, if RC5 [26] is used for providing all these security primitives, the total number of RC5 computations is about $log(m) + 3$.

  Now let us consider the computational cost for computing a pairwise key based on the Blundo scheme. Let the degree of a polynomial share be t, the size of a coefficient in a polynomial be 64 bits, and the sizes of a regular node id and a MS node id be 16 bits and 64 bits, respectively. For a MS node, it evaluates the id of a regular node at its polynomial share. The number of modular multiplications is $t + 1$ and every modular multiplication is between a 64-bit number and a 16-bit number. For a host node, it evaluates the id of a MS at its polynomial share. The number of modular multiplications is also $t + 1$, but every modular multiplication is between two 64-bit numbers. Hence, the computational overhead for a host node in computing a pairwise key is 4 times as large as that for a MS node. It has been shown [11] that when $t = 50$ and an id is 16 bits, the number of CPU cycles for computing a pairwise key based on the Blundo scheme is equivalent to that for computing a RC5 MAC; therefore, when $t = 50$, the computational cost on a host node side is equivalent to computing four RC5 MACs.

  Combining the computational costs in all the above processes. When $m = 64$ and $t = 50$, the overall computational cost taken by a host node is equivalent to $log(m) + 3 + 4 = 13$ RC5 MACs. It has been shown that the energy a sensor node spends on computing one MAC is about the same as that used for transmitting one byte [27]. Thus, from energy point of view, the energy used by a host node to establish a pairwise key with a MS node is about the same as that used in transmitting 13 bytes. The computation cost of a MS is smaller because it does not need to compute hashes. In practice, we believe this is an affordable overhead for the current generation of sensor nodes.

- *Communication Cost* The authentication between a MS and a host node involves totally three messages. The first message sent from the MS to a host node includes TT, T_e, T_s, MS and $log(m)$ auxiliary values in the tree, where m is the number of blocks to be traversed by the MS. The two authentication messages are very small because they only include a nonce and a MAC.

- *Storage Overhead* A regular sensor node only needs to store its polynomial share, i.e., $(t + 1)$ coefficients. In addition to storing a polynomial share, a MS stores all the block ids. As an example, suppose a sensor field is divided into 64X64

cells and the maximum size of a block is 8 cells, then we can use two bytes to represent a block id. Hence, even if a MS is to traverse hundreds of blocks and its resources are as scarce as those of a Mica2-based mobile sensor [18], the storage overhead is still not a concern.

3.2 Revoking a Mobile Sink On-Demand

If a mobile sink (MS) is detected compromised when its granted privilege is still valid, the MS should be revoked as soon as possible. To do this, the base station may send a revocation message to all the host nodes of the MS. As a simple approach, the base station may send a revocation message to each host node individually. However, this is not feasible since the base station may not know the ids of these nodes. Even the ids are known, the communication overhead increases rapidly as the number of host nodes increases. Alternatively, the base station may flood the revocation message over the network. This is still not efficient because all nodes are involved in receiving or forwarding the message. For efficiency, a revocation message should be multicast only within the smallest area (called *revocation area*) that covers the host nodes of the MS.

3.2.1 Basic Scheme

To multicast a revocation message within the revocation area, the basic idea of location-based multicasting [28, 29] can be applied. The base station first generates a message which indicates the id of the MS to be revoked and the scope of revocation area. Then, the base station broadcasts the message to its neighbors. On receiving the message, each node acts as follows:

- If it has received the message before or is outside of the revocation area, the message is dropped.
- If it is within the revocation area indicated by the message, it records the id of the revoked MS, and rebroadcasts the message to its neighbors.

The basic scheme is efficient when the revocation area is regular; for example, it is a rectangle or a circle. In many scenarios, however, the revocation area could be in any arbitrary shape. Although an irregular area can be divided into and represented as a set of smaller regular (e.g., rectangular) subareas (as described in Section 3.1.3), this may need large space in a revocation message. For example, if a revocation area is represented as 100 rectangles, and each rectangle needs 4 bytes, 400 bytes are demanded to represent the area. This is not trivial for the current sensor network in which a packet contains only a few tens of bytes. To revoke the MS, $\lceil \frac{400}{29} \rceil = 14$ revocation messages must be multicast to the host nodes. In order to broadcast all these messages, each host node needs to record the forwarding path when receiving the first message for this revocation event. Based on this information, it can forward the following revocation messages received later. During this course, each host node has to receive and/or forward 14 packets.

3.2.2 Enhanced Schemes

To address the problems of the basic scheme, we use two techniques:

(1) The revocation area is divided into multiple subareas, and multiple revocation messages are sent to and multicast within these subareas simultaneously. Using this technique, the revocation delay can be reduced.

(2) The basic blocks forming the revocation area are further combined into a smaller number of blocks (called *expanded blocks*). Using this technique, the number of revocation messages can be reduced.

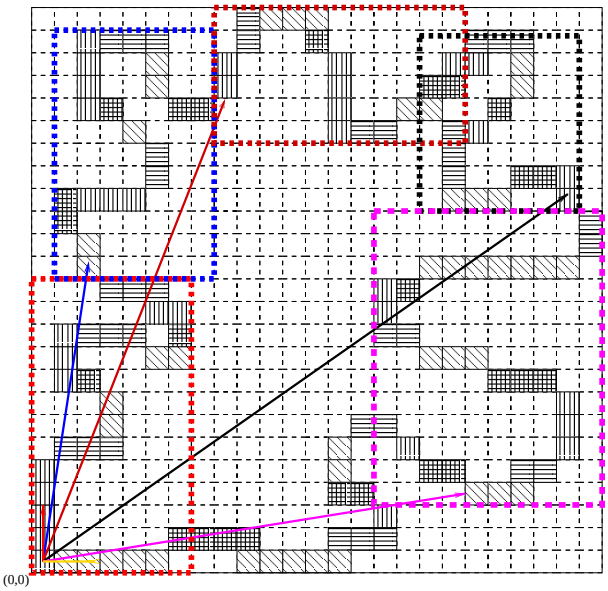

Figure 5: Illustration of the GPSR-based scheme

In the following, we first present two enhanced schemes, each is based on one of the above techniques. Then, we use both techniques to design another enhanced scheme.

OPT-I: GPSR-based Scheme When the revocation area is large or complicated, and it must be represented using multiple revocation messages, we may not solely rely on host nodes to forward these message. Instead, we can use the GPSR protocol [30, 31, 32] to send each revocation message to a certain node within the subarea indicated by the message, and then multicast the message within that subarea.

Figure 5 illustrates how to multicast revocation messages using the GPSR-based scheme. Unlike in the basic scheme all the revocation packets follow the trajectory, in the GPSR-based scheme each packet is forwarded towards the first cell of its destination area, based on a path dynamically determined by the GPSR routing protocol. Once a revocation packet arrives at its destination area, it is multicast and forwarded based on the block ids contained in the packet. For illustration purpose, in the figure we plot the GPSR paths as straight lines, although in reality they could be more complex subject to factors such as node distribution and network topology.

An important advantage of the GPSR-based scheme over the basic scheme is that it can significantly reduce the revocation latency. This is because the base station can send multiple packets almost simultaneously along different paths and every path is the shortest path between the base station and its destination revocation area. In the basic scheme, multiple packets follow the direction of the trajectory, thus the sensor nodes within the cells (blocks) that are in the arriving direction to the base station in the trajectory always receive the revocation notification at the latest. In practice, we should have the high priority to notify these sensor nodes because it is likely a compromised MS has not contacted with them yet. Thus using the GPSR-based scheme allows us to notify each subarea of host nodes as early as possible.

On the other hand, the GRSR-based scheme introduces some additional overhead. When a revocation message is sent along a path towards a revocation area, all the nodes in the cells cross the path are involved in receiving and/or forwarding the message. We call these cells *redundant cells*. The cost of sending a revocation message can be measured by the number of redundant cells cross the path along which the messages is transmitted.

OPT-II: Minimum Message Scheme Suppose the original revocation area is represented by b rectangular blocks (called *basic blocks*) using the algorithm presented in Section 3.1.3, and one packet can contain the representation of at most k ($k < b$) blocks. To reduce the number of revocation messages, we can further combine the basic blocks into a limited number (say, k) of larger rectangular blocks (called *expanded blocks*), such that only a minimal number of revocation messages are demanded. Note that, an expanded block can also be represented as a four-variable tuple (i, j, l_x, l_y), where (i, j) is the index of the starting cell, and l_x (l_y) is the length (in the unit of cell) of the block in the X (Y) dimension. So the representation of an expanded block has the same size as that of a basic block. During this expanding course, some cells that are not included in any basic block may be included in some expanded block. We also call these cells *redundant cells*. Multicasting a revocation message to the redundant cells is not harmful, but it increases communication overhead. Therefore, the number of redundant cells should be minimized. In the following, we present a dynamic programing-based approach to minimize the number.

- According to the visiting order, all the basic blocks are sorted into a sequence denoted as follows:

  $$B_1, B_2, \cdots, B_b.$$

  Here, b is the total number of basic blocks, and block B_i is visited earlier than any block B_j ($j > i$).

- Let $R_m(i, l)$ be the minimum number of redundant cells introduced when basic blocks $B_i, B_{i+1}, \cdots, B_b$ are combined into at most l expanded blocks, and $R(i, j)$ be the minimum number of redundant cells introduced when basic blocks B_i, $\cdots$, B_j are combined into a single expanded block. Then, the minimum number of redundant cells introduced when all the basic blocks into at most k expanded blocks, denoted as $R_m(1, k)$, can be computed as follows:

  $$R_m(i,l) = \begin{cases} 0 & i = b, \\ R(i,b) & i = 1, \\ min\{R(i,j) + R_m(j+1, l-1) \\ \mid i \le j \le b - l + 1\} & \text{others.} \end{cases} \quad (11)$$

  Here, $1 \le i < b$ and $2 \le l \le k$.

A more detailed description of the algorithm is shown in Figure 6. Figure 7 shows an example, in which the revocation area is represented by 20 expanded blocks and two messages are used to revoke all the host nodes.

OPT-III: Optimized Multi-message Scheme The above enhanced schemes still have some limitations, especially when the revocation area is composed of a large number of basic blocks. In the GPSR-based scheme, we may need many revocation messages to represent the whole revocation area; i.e., the base station may have to send many revocation messages. The minimum message scheme can minimize the number of revocation messages, but lots of redundant cells may be introduced. Since the overall communication cost increases as the number of redundant cells or revocation messages increases, it is important to carefully design the representation scheme to minimize the overall cost.

Notations

- B_i, $R_m(i, l)$, $R(i, j)$: defined before.
- $next(i, l)$: when basic blocks $B_i, \cdots, B_b$ are combined into at most l expanded blocks, the first expanded block is composed of $B_i, \cdots, B_{next(i,l)-1}$; i.e., $next(i, l)$ is the starting cell of the next expanded block.

The Algorithm

```
for l = 1 to k
  R_m(b+1, l) = 0
for i = 1 to b
  R_m(i, 1) = R(i, b) /*initialization*/

for i = b to 1
  for l = 2 to k
    R_m(i, l) = +∞
    for j = i to b − l + 1
      if R(i, j) + R_m(j + 1, l − 1) < R_m(i, l) then
        R_m(i, l) = R(i, j) + R_m(j + 1, l − 1);
        next(i, l) = j + 1

/*Following:
  output the starting cell of each expanded block*/
i = 1; print i
for l = k to 1
  if next(i, l) > b then end
  else print next(i, l)
  i = next(i, l)
```

Figure 6: The algorithm for the minimum message scheme

In the following, we extend the algorithm described in Figure 6 to design a new algorithm that can minimize the overall communication cost. We consider the overall communication cost including three parts:

- the necessary cost of multicasting a revocation message within the original revocation area;
- the additional cost of multicasting the message within the redundant cells; and
- the additional cost of sending revocation messages from the base station to all the sub-scopes.

In the algorithm, we use the following notations:

- k,b: as in the previous algorithm, k is the maximum number of expanded blocks that can be represented in a single packet, and b is the maximum number of basic blocks in the whole revocation area.
- $R_m(i, l, e)$: this is extended from the notation $R_m(i, l)$ defined in the previous algorithm. It represents the minimum number of redundant cells introduced when basic blocks B_i, B_{i+1}, $\cdots$, B_e are combined into at most l expanded blocks.
- $C(i)$: the number of cells involved in sending a revocation message from the base station to block B_i.

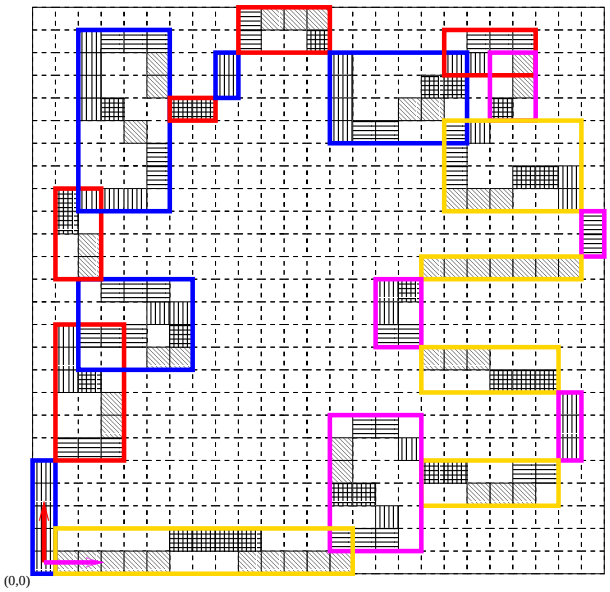

Figure 7: Illustration of the minimum message scheme

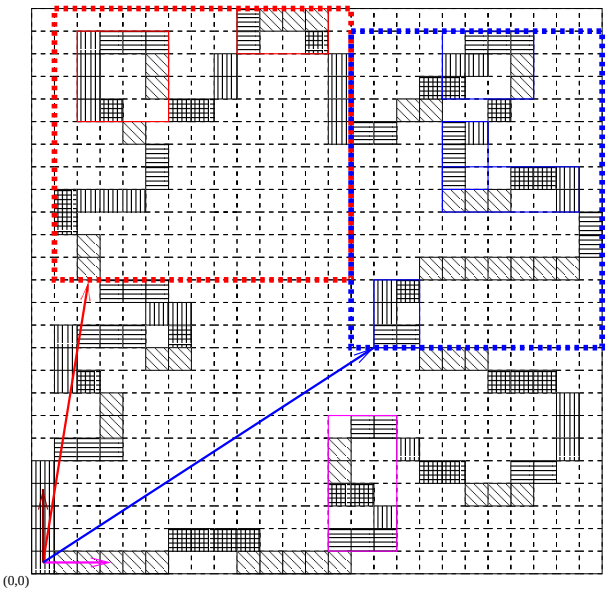

Figure 8: Illustration of the optimized multi-message scheme

- $C_m(i)$: the minimum number of redundant cells introduced when sending revocation messages to host nodes in blocks $B_i, \cdots, B_b$.

Using the dynamic programming technique, $C_m(i)$ can be calculated as follows:

$$C_m(i) = \begin{cases} C(i) & i = b, \\ min\{R_m(i,k,j) + C(i) + C_m(j) \\ \quad | \; i \leq j \leq b - l + 1\} & \text{others.} \end{cases} \quad (12)$$

Figure 8 shows the result of applying the optimized multiple message scheme on the previous example. After using this scheme, the 57 basic blocks are combined into 40 expanded blocks, and 4 revocation messages are used to revoke the MS.

3.2.3 *Discussions*

During the description of all the four revocation schemes, we have implicitly assumed that a revocation packet can arrive at the start forwarding cell reliably and correctly. If the adversary has compromised some stationary sensor nodes, these nodes may attack the revocation schemes by dropping the revocation messages that they should forward, modifying passing revocation messages, or inserting false revocation messages.

To prevent compromised nodes from modifying revocation messages or inserting false revocation messages, the messages should be authenticated. For example, we may use the μTESLA scheme [33] for this purpose. Packet dropping cannot be prevented, but may be detected. The *watchdog* mechanism [34] and the reputation-based scheme [35] may be employed to thwart a compromised nodes from dropping revocation messages.

3.3 Performance Evaluations

We evaluate the performance of the revocation algorithms by simulations. We assume that a revocation packet can only store ten blocks. As a result, multiple revocation packets are needed when the number of blocks are larger than ten. Four revocation algorithms will be evaluated: the Basic algorithm, the GPSR-based scheme(OPT-I), the minimum message scheme(OPT-II), and the optimized multi-message scheme (OPT-III). All these four schemes have been discussed in the last section.

Three metrics are used to evaluate the performance of the proposed schemes: the *average revocation delay*, the *maximum revocation delay*, and the message overhead. The revocation delay for a cell is defined as the number of hops that the revocation message transmits before it reaches the cell. The average revocation delay is the average of all revocation delays for all cells on the trajectory. Similarly, the maximum revocation delay is the maximum among all the delays. The message overhead is defined as the total number of transmitted hops of all the revocation messages. In OPT-II and OPT-III, the message overhead also includes the messages due to using redundant cells.

These four algorithms are evaluated using four typical trajectories: triangle, polygon, ellipse and irregular shape, and the results are illustrated in Figures 9, 10, 11 and 12, respectively.

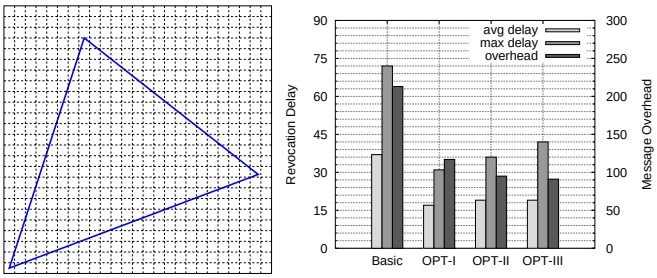

Figure 9: Triangle Trajectory

From the figures, we can make the following observations. First, all three optimization schemes outperform the basic scheme. In the basic scheme, all the revocation messages are sent in one direction along the trajectory, resulting in a large revocation delay and high message overhead.

Second, in terms of revocation delay, the OPT-I scheme performs the best. In the OPT-I scheme, the revocation messages are sent in both directions simultaneously. Instead of following the trajectory, each message is forwarded to the first block in the message using

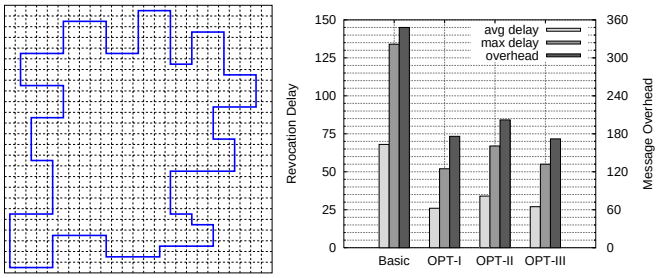

Figure 10: Polygon Trajectory

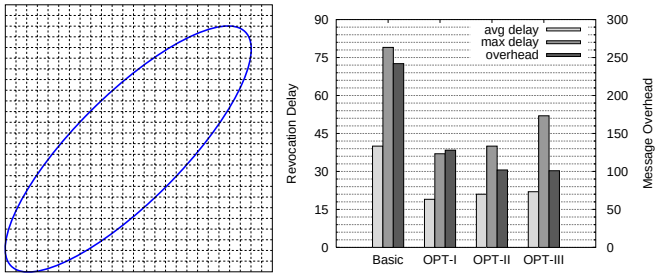

Figure 11: Ellipse Trajectory

the shortest path, thus reducing the revocation delay. In addition, the number of messages are larger, or at least equal to, the other two optimized schemes, which results in smaller number of blocks in each message. The smaller the number of blocks in a message, the faster the message can be forwarded to them. All these explain why OPT-I has the lowest revocation delay among all the schemes.

Third, in terms of message overhead, the OPT-III scheme performs the best. Both OPT-II and OPT-III are designed to minimize the message overhead. As a result, both outperforms OPT-I in terms of message overhead. Furthermore, the OPT-III scheme is designed to minimize the message overhead using multiple messages. Therefore, OPT-III further reduces the message overhead compared to OPT-II.

Fourth, from Figures 10 and 12, we can see that OPT-II does not perform very well compared to OPT-I and OPT-III when the trajectory is complex. This can be explained as follows. In the OPT-II scheme, no matter how many blocks exist, the scheme always ends up with two revocation messages. Hence, the revocation delay is high when the trajectory is complex and the number of blocks is high. The message overhead is also increased due to the use of redundant cells.

In summary, there is a tradeoff between revocation delay and message overhead. In practice, we can choose different revocation schemes based on the design goals and the requirements of the applications. For example, if the design goal is to revoke the MS faster, we can use the OPT-I scheme. On the other hand, if the design goal is to revoke the MS with minimum message overhead, we can choose the OPT-III scheme.

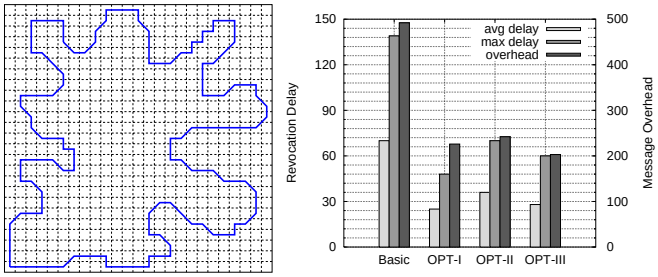

Figure 12: Irregular Shape Trajectory

4. RELATED WORK

Establishing pairwise keys between two regular sensor nodes has been extensively studied recently. There are schemes using a trusted third party (base station) [7], schemes exploiting the initial trustworthiness of newly deployed sensors [12], and schemes based on the framework of probabilistic key predeployment [8, 9, 10, 11, 36]. The Blundo scheme [24], which is a threshold-based pairwise key establishment scheme, has been recently extended to enable a sensor network to sustain more node compromises under the same memory constraints [9, 11, 37]. Our schemes can also be constructed from these extended schemes if necessary although we used the Blundo scheme.

Perrig *et al.* [7] presented μTESLA for base station broadcast authentication, based on one-way key chains and delayed key disclosure. Liu and Ning [6] proposed several multilevel μTESLA schemes to further extend the scalability of the original μTESLA. Zhu et al [12] and Deng *et al.* [38] presented several local one-hop or multiple-hop broadcast authentication schemes that are also based on one-way key chains. The latter schemes allow a receiver node to verify a broadcast message immediately but with weaker security than the μTESLA-based schemes. Recently, Zhu *et al.* [39] and Ye *et al.* [40] proposed data authentication schemes, which can filter false sensor data with high probability. We note that these schemes only provide source authentication, thus they cannot be applied when the communication between a MS and a host node should be confidential. Moreover, these schemes cannot restrict the privilege of a MS if employed directly.

Wood and Stankovic [14] identified a number of DOS attacks in sensor networks. Deng *et al.* [15] proposed a multiple-base station and multiple-path strategy to increase intrusion tolerance, and an anti-traffic analysis strategy to disguise the location of a base station. Karlof and Wagner [13] described several security attacks on routing protocols for sensor networks. The attacks that are difficult to detect or prevent is the one that combines the *Sinkhole* and the *Wormhole* attacks [41]. These attacks however do not directly apply to our schemes. Zhang *et al.* [42] proposed predistrition and local collaboration-based group rekeying schemes to revoke compromised nodes. These schemes assume that the compromised nodes are stationary, so they cannot be directly applied to this work.

Location-based multicasting (or *geocasting*) has been studied in the environments of mobile ad hoc networks [28] and sensor networks [29]. In these schemes, a source node sends out a packet in which a predicted destination multicasting scope is specified as a regular area (e.g., a circle or a rectangle). The packet is first forwarded toward the destination area, using a location-based routing protocol, and then is flooded within that area. These schemes cannot be directly applied to our revocation problem since they assume a regular multicasting area, while the revocation area could be any irregular shape. Due to the irregularity, it may takes large space to represent the revocation area. This is not trivial for the current generation of sensor networks, in which the packet size is only a few tens of bytes. So, several techniques are adopted by our schemes to simplify the representation, and hence reduce the multicasting overhead. In addition, the multicasting area in the previous schemes cannot be predetermined accurately, while the revocation area in our schemes can be pre-specified, which allows the optimizations proposed in this paper.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we identified new security challenges of using MSs; if a MS is given too many privileges, the compromise of the MS may break down the whole network; on the other hand, without some necessary privilege, the MS may not be able to accomplish the planned mission. Based on the principle of least privilege, we first proposed several efficient schemes to restrict the privilege of the MS without impeding its capability of carrying out any authorized operations for an assigned task. To further reduce the possible damage caused by a compromised MS, we then proposed efficient message forwarding schemes for depriving the privilege assigned to the compromised MS immediately after its compromise has been detected. Detailed analysis and simulation results show that our schemes are secure and efficient, and are highly practical for sensor networks consisting of the current generation of sensors.

To the best of our knowledge, this is the first paper to address the MS privilege issues in wireless sensor networks. As the initial work, we do not expect to solve all the problems. In the future, we will address issues when the MS needs to change its trajectory due to unexpected events. We will also look into other revocation techniques to balance the tradeoff between delay and message complexity.

Acknowledgments

We would like to thank Wenliang Du for his insightful comments and suggestions. We would also like to thank the anonymous reviewers for their helpful suggestions. This work was supported in part by the National Science Foundation (CNS-0092770 and ITR-0219711).

6. REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, March 2002.

[2] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," *ACM MobiSYS'04*, June 2004.

[3] Y. Tirta, Z. Li, Y. Lu, and S. Bagchi, "Efficient Collection of Sensor Data in Remote Fields Using Mobile Collectors," *The 13th International Conference on Computer Communications and Networks (ICCCN 2004)*, October 2004.

[4] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *ACM International Conference on Mobile Computing and Networking (MOBICOM'02)*, pp. 148–159, September 2002.

[5] W. Zhang, G. Cao, and T. La Porta, "Data Dissemination with Ring-Based Index for Wireless Sensor Networks," *IEEE International Conference on Network Protocols (ICNP)*, pp. 305–314, November 2003.

[6] D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," in *Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS'03)*, 2003, pp. 263–276.

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

[8] H. Chan, A. Perrig, D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. of the IEEE Security and Privacy Symposim 2003*, May 2003.

[9] W. Du, J. Deng, Y. Han, and P. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003, pp. 42–51.

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

[11] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, 2003, pp. 52–61.

[12] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, 2003, pp. 62–72.

[13] C. Karlof and D. Wagner, "Secure Routing in Sensor Networks: Attacks and Countermeasures," *Proc. of First IEEE Workshop on Sensor Network Protocols and Applications*, May 2003.

[14] A. Wood and J. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, pp. 54–62, 2002.

[15] J. Deng, R. Han, and S. Mishra, "Intrusion tolerance strategies in wireless sensor networks," in *Proceedings of IEEE 2004 International Conference on Dependable Systems and Networks (DSN'04)*, 2004.

[16] F. Stajano and R. Anderson, "The protection of information in computing systems," in *Proceedings of the IEEE*, 1975.

[17] CROSSBOW TECHNOLOGY INC., "Wireless sensor networks," *http://www.xbow.com/Products/Wireless_Sensor_Networks.htm*.

[18] UC Berkeley The EECS department, "Cotsbots: The mobile mote-based robots," *http://www-bsac.eecs.berkeley.edu/projects/cotsbots/*.

[19] Y. Xu, J. Heidemann and D. Estrin, "Geography Informed Energy Conservation for Ad Hoc Routing," *ACM MOBICOM'01*, July 2001.

[20] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of ACM SenSys'03*, 2003.

[21] S. Capkun and J. Hubaux, "Secure positioning in sensor networks," in *Technical report EPFL/IC/200444*.

[22] L. Lazos and R. Pooverdran, "Serloc: Secure range-independent localization for wireless sensor networks," in *Proceedings of of ACM Workshop WiSe'04*, 2004.

[23] O. Goldreich, S. Goldwasser, and S. Micali, "How to Construct Random Functions," *Journal of the ACM*, vol. 33, no. 4, pp. 210–217, 1986.

[24] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung, "Perfectly-secure key distribution for dynamic conferences," in *Advances in Cryptology, Proceedings of CRYPTO'92*, 1993, LNCS 740, pp. 471–486.

[25] Ralph Merkle, "A certified digital signature," in *Proceedings of Advances in Crypto-89*, 1989, pp. 218–238.

[26] R. Rivest, "The rc5 encryption algorithm," in *Proceedings of the 1st International Workshop on Fast Software Encryption*, 1994, pp. 86–96.

[27] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route detection and filtering of injected false data in sensor networks," in *Proceedings of IEEE Infocom'04*, 2004.

[28] Y. Ko and N. Vaidya, "GeoTORA: A Protocol for Geocasting in Mobile Ad Hoc Networks," *International Conference on Network Protocols (ICNP)*, November 2000.

[29] Q. Huang, C. Lu, and G. Roman, "Spatiotemporal Multicast in Sensor Networks," *ACM Sensys'03*, November 2003.

[30] I. Stojmenovic P. Bose, P. Morin and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *International Workshop on Discrete Algorithms and methods for mobile computing and communications.*

[31] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *The Sixth Aunual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)*, Aug. 2000.

[32] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," *ACM MobiHoc'03*, 2003.

[33] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "SPINS: Security Suite for Sensor networks," *Mobicom'01*, 2001.

[34] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *ACM MobiCom*, August 2000.

[35] S. Ganeriwal and M. Srivastava, "Reputation-based framework for high integrity sensor networks," in *Proceedings of ACM Workshop on the Security of Ad Hoc and Sensor Networks (SASN'04)*, 2004.

[36] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing Pairwise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach," *IEEE International Conference on Network Protocol (ICNP)*, November 2003.

[37] D. Medhi D. Huang, M. Mehta and L. Harn, "Location-aware key management scheme for wireless sensor networks," in *Proceedings of Workshop on Security of Ad Hoc and Sensor Networks*, 2004.

[38] J. Deng, R. Han, and S. Mishra, "Security support for in-network processing in wireless sensor networks," in *Proceedings of First ACM Workshop on the Security of Ad Hoc and Sensor Networks (SASN'03)*, 2003.

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

[40] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical En-route Filtering of Injected False Data in Sensor Networks," *IEEE INFOCOM'04*, March 2004.

[41] Y. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks," *Proceedings of INFOCOM 2003*, April 2003.

[42] W. Zhang and G. Cao, "Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration based approach," *IEEE INFOCOM'05*, 2005.