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

HUI SONG
Frostburg State University
SENCUN ZHU
The Pennsylvania State University
WENSHENG ZHANG
Iowa State University
and
GUOHONG CAO
The Pennsylvania State University

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 article, based on the *principle of least privilege*, we first propose an efficient scheme to restrict the privilege of a mobile sink without impeding its ability to carry out any authorized operations for an assigned task. In addition, we present an extension to allow conditional trajectory change due to unexpected events. To further reduce the possible damage caused by a compromised mobile sink, we propose efficient message forwarding schemes for deleting the privilege assigned to a compromised mobile sink immediately after its compromise has been detected. Through detailed

This work was supported in part by the Army Research Office (W911NF-05-1-0270, MURI W911NF-07-1-0318) and the National Science Foundation (CNS-0524156, CNS-0519460, and CAREER-0643906). A preliminary version of this article appeared in Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05), Urbana-Champaign, IL. May 25–28, 2005.
Authors' addresses: H. Song, Department of Computer Science, Frostburg State University, Frostburg, MD 21532; email: hsong@frostburg.edu; S. Zhu and G. Cao, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802; email: {szhu, gcao}@cse.psu.edu; W. Zhang, Department of Computer Science, Iowa State University, Ames, IA 50011; email: wzhang@cs.iastate.edu.

analysis, simulation, and real implementation, 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.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; D.4.6 [**Operating Systems**]: Security and Protection—*Cryptographic controls*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Security, Algorithms, Design

Additional Key Words and Phrases: Least privilege, key revocation, key management, pairwise key, sensor networks

ACM Reference Format:

Song, H., Zhu, S., Zhang, W., and Cao, G. 2008. Least privilege and privilege deprivation: toward tolerating mobile sink compromises in wireless sensor networks. ACM Trans. Sens. Netw., 4, 4, Article 23 (August 2008), 34 pages. DOI = 10.1145/1387663.1387669 http://doi.acm.org/10.1145/1387663.1387669

1. INTRODUCTION

Mobile sinks (or mobile soldiers, or mobile sensors, as shown in Figure 1) are essential components for the operation of many sensor network applications. One such application is data collection. Wireless sensor networks [Akyildiz et al. 2002] allow continuous monitoring of the environment in hazardus 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 a 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 [Kansal et al. 2004; Tirta et al. 2004] have proposed temporarily storing the data in sensor nodes, and letting the base station periodically dispatch mobile sinks (MSs) to collect data. In some strategic scenarios such as battlefield, MSs [Ye et al. 2002; Zhang et al. 2007] 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 an MS however, introduces a new security challenge: the privilege granted to an MS can be abused once it is compromised. This will become a serious problem if an MS is granted many privileges, making it very attractive for attack and compromise. For example, suppose we want to send an MS to collect sensor readings in a specific location during a specific time interval. Without appropriate restrictions, a compromised MS may be able to collect sensor data from the entire network at all times. In another example, suppose we want to dispatch an 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 as well as, which highlights

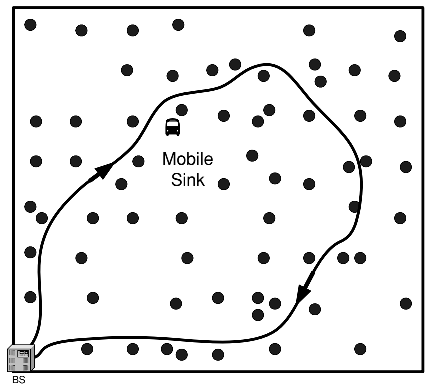


Fig. 1. An example application where an MS is dispatched to carry out a task along a predetermined trajectory.

indicates the importance of restricting the privileges of MSs. On the other hand, the convenience of an MS in executing the authorized operations should not be sacrificed by the increased security restrictions.

We can limit the privilege of an 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 [Liu and Ning 2003a; Perrig et al. 2001], key management [Chan et al. 2003; Du et al. 2003; Eschenauer and Gligor 2002; Liu and Ning 2003b; Zhu et al. 2003a], and others [Karlof and Wagner 2003; Wood and Stankovic 2002; Deng et al. 2004]. These schemes may be employed for securing communication between regular sensor nodes or between an MS and a regular sensor, but they cannot restrict the privilege of an MS while giving enough privileges 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* [Saltzer and Schroeder 1975], we design a security restriction scheme that only grants the MSs the minimum privilege required to accomplish their tasks. This scheme allows and only allows an MS to perform the pre authorized operations. A salient feature of our scheme 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 an MS. Indeed, our constructions guarantee that (1) an MS cannot lie or modify its assigned task, and (2) 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 and implementation that this scheme is very effective and efficient. In addition, we present an extension to allow conditional trajectory change due to unexpected events.

Second, we propose several on-demand schemes for privilege deprivation: a basic scheme followed by three optimized schemes that can revoke the privilege granted to the MS. Privilege deprivation is important and necessary when an 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 proposed privilege restriction scheme and one extension to it. Section 4 presents four privilege revocation schemes. Section 5 presents our real implementation of the proposed privilege restriction scheme in MICA2 motes. Section 6 describes some related work. Finally, Section 7 concludes the article 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 the current generation of sensor nodes, such as MICA/MICA2 motes developed at UC Berkeley [Crossbow Technology Inc. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm]. 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 that have the same amount of resources as the MICA2 Motes, such as CotsBots [Bergbreiter and Pister 2003], Micabot [McMickell et al. 2003], and Robomote [Sibley et al. 2002], to name a few. 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 [Xu et al. 2001]. We consider the type of applications in which we dispatch an MS with a known purpose. More specifically, an MS is assumed to perform some tasks along a roughly predetermined physical path (perhaps an arbitrary curve) in a sensor network. We know the type of task (e.g., collecting data or network diagnosis), the start time and the end time, and the cells involved in the task (the last assumption is relaxed in Section 3.4). 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 [Zhang and Cao 2004], data aggregation, TDMA radio scheduling, and so on. It is also needed in our scheme since a sensor node needs to verify if the task claimed by an MS has the valid time interval.

The required accuracies for both localization and time are application dependent. Currently, using the state of art localization schemes [Priyantha et al. 2000; Savvides et al. 2001], one can achieve meter- or even centimeter-level accuracy; on the other hand, existing time synchronization protocols such as Ganeriwal et al. [2003]; Song et al. [2007]; and Sun et al. [2006] are shown to be able to provide millisecond- or even microsecond-level precision. Generally, the errors introduced by both localization and time synchronization are tolerable in our system.

2.2 Security Assumptions

We assume that the base station has a mechanism to authenticate broadcast messages (e.g., based on μTESLA [Perrig et al. 2001]), and every node can verify the broadcast messages. We also assume that when an adversary compromises a node, either a regular node or an 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. Attacks and countermeasures for node localization schemes and time synchronization have been presented in Capkun and Hubaux [2002]; Lazos and Poovendran [2005]; Liu et al. [2005]; and Capkun et al. [2006] and in Anjum et al. [2005]; Song et al. [2007]; Ganeriwal et al. [2005]; and Sun et al. [2006], respectively.

The motivation of privilege deprivation (interchangeably called MS revocation) can be due to the change of security policy, the detection of MS compromises, or other reasons. In particular, we do not assume that the reason for an 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, an MS revocation is often the result of revoking the carrier of the MS. For example, if a mobile soldier is captured by the adversary or is missing in a battlefield, other soldiers can report the event to the network controller, which initiates an MS revocation operation to revoke the MSs carried by this soldier.

2.3 Design Goal

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

—*Least privilege.* An 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 securely communicate with sensor nodes.

—*Immediate privilege deprivation.* Once the compromise of an 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 the resource-constrained sensor nodes and MSs, the schemes should be efficient in terms of communication, computation, and storage.

2.4 Notations

The following notations appear in the rest of this discussion:

—u (lower case) is a regular stationary sensor node;

—MS is a mobile sink;

—TT is the type of 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 an MS if the MS is authorized to talk to it in a task.

3. RESTRICTING THE PRIVILEGE OF A MOBILE SINK

To restrict the privilege of an 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 an MS can only carry out its authorized task. As an 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 damage to the host nodes.

We propose two sets of solutions to address the MS compromise problem in the following two sections. This section discusses our proposed schemes for restricting the privilege of an MS. Section 4 presents four message forwarding schemes (a basic scheme followed by three optimization schemes) for efficiently and quickly delivering an MS revocation notification to all the host nodes of the compromised MS.

3.1 The Strawman Scheme

In the strawman scheme, every node is preloaded 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) [Goldreich et al. 1986]. 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.

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

where H is a collision-resistant one-way hash function and $\mid$ 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 each other by, for example, exchanging the following authentication messages

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

$$u \rightarrow MS : u, seq + 1, MAC(K_u(MS), u \mid seq + 1). \tag{3}$$

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. In addition, node u saves MS (the ID of the mobile sink) and seq if the verification for message (2) succeeds. 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 MS maintains the seq, which is increased by one each time the MS communicates with a new node. Here seq is used for preventing replay attacks. Specifically, by saving MS and seq at node u, future authentication messages like (2), claimed from the same MS, will be dropped automatically. Similarly, replayed message (3) will be dropped by the MS automatically also. On the other hand, because of saving MS and seq information at node u, the storage overhead for u could be large if there are a huge number of legitimate MSs in the system. In this case, a window-based strategy will help make a tradeoff.

The strawman scheme meets our design goal and should work well if an 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 troubling limitation is the requirement 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 (nodes located on the trajectory).

Thus, the BS cannot preload the MS with the pairwise keys shared with the en route host nodes.

Next, we present our proposed scheme, which is built on the strawman scheme and provides scalability and security while restricting the privilege of the MS.

3.2 The Proposed Scheme

The limitations of the strawman scheme are due to the lack of scalability of the PRF-based pairwise key predeployment scheme. In this subsection, we first present a Blundo scheme-based construction to achieve scalability. Then we use techniques such as Merkle hash tree to address the security concerns remaining in the Blundo scheme-based construction.

3.2.1 *A Blundo Scheme-Based Construction.* 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 that allows an MS to establish a pairwise key with any sensor node, on the fly.

Recently many pairwise key establishing schemes [Chan et al. 2003; Du et al. 2003; Eschenauer and Gligor 2002; Liu and Ning 2003b; Zhu et al. 2003a] 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 [Blundo et al. 1993] to construct our protocols, although several other schemes [Du et al. 2003; Liu and Ning 2003b] might be adapted for our purpose as well. As we shall see shortly, the Blundo scheme provides clear security guarantees. 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 n with $f(n, y)$, which is a polynomial obtained by evaluating $f(x, y)$ at $x = n$.

—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 ($f(MS, y)$), 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 limitations in the strawman scheme.

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 bivariate 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 [Liu and Ning 2003a; Du et al. 2003].

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

$$MS(u) = H(TT \mid T_s \mid T_e \mid u). \tag{5}$$

The BS then preloads 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 the strawman scheme.

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.

3.2.2 *Reducing The Number of Polynomial Shares To One.* To address the remaining issues of the Blundo-based construction, we reduce the number of polynomial shares possessed by an MS—ideally an 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 an MS. Second, we use a Merkle [1989] hash tree to construct the ID for an MS, so that only one polynomial share has to be assigned to the MS. In the following, 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 3. If the MS is scheduled to cross cell (i, j), BS can generate a specific ID for MS, $MS(i, j)$:

$$MS(i, j) = H(TT \mid T_s \mid T_e \mid i \mid j). \qquad (6)$$

Now the MS can establish a pairwise key $f(MS(i, j), u)$ with any node u in the cell (i, j) using the Blundo scheme-based technique. 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 an 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 an 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 1 bit. $s = 0$ means that a block has only one cell, the base cell. In Figure 3, 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 ith row and the ith column is processed in the ith iteration. We find the first horizontal block in the ith row and the first vertical block in the ith 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 2. Figure 3 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. Suppose we have obtained m blocks after running this cell-merging algorithm. Let the ID of block i be B_i. A Merkle hash tree is constructed 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 $X_{18} = H(X_{14} \mid X_{58})$, $X_{14} = H(X_{12} \mid X_{34})$, $X_{34} = H(B_3 \mid B_4)$, and H is a

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 $\mathcal{B}$
        $j = j + 1$
```

Fig. 2. The basic algorithm for finding blocks.

collision-resistant hash function. we can derive the ID of the MS node as:

$$MS = H(TT \mid Ts \mid Te \mid X_{1m}). \tag{7}$$

To establish a pairwise key with a node u in block B_i, the MS provides MS, TT, Ts, Te, and 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 $X_{18} = H(H(X_{12}|H(B_3|B_4))|X_{58})$; otherwise, it terminates the verification process. Next, node u derives the ID MS based on Equation (7) and further computes its pairwise key $f(u, MS)$ shared with the MS. The MS also computes $f(MS, u)$. Finally, as in the strawman scheme, node u and MS provide mutual authentication using their pairwise key as the MAC key.

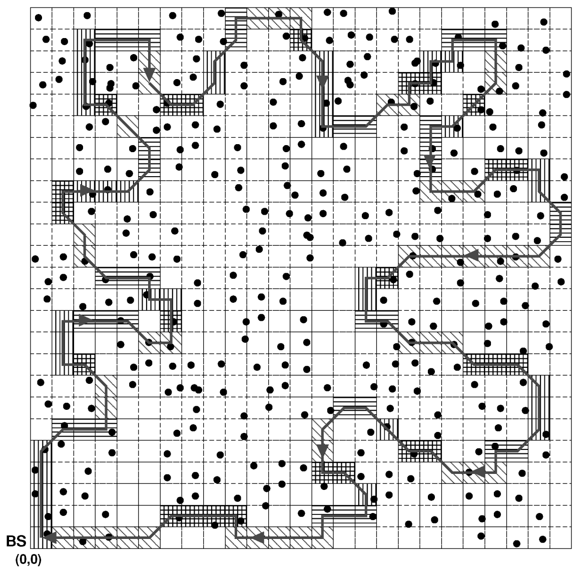


Fig. 3. A sensor network field is divided into cells and a mobile sink traverses the field along a predetermined trajectory.

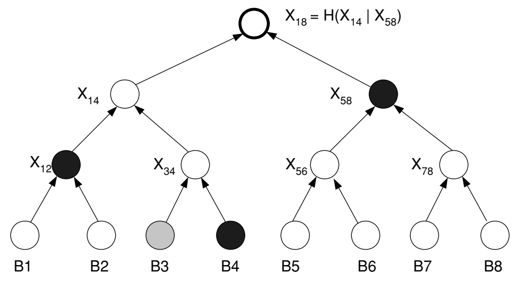


Fig. 4. A Merkle-hash tree constructed from the IDs of the blocks to be traversed by a mobile sink.

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 $X_{18} = H(H(X_{14}|(H(B_5|B_6)|X_{78}))$. 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.

3.2.3 *Defending Against Denial-of-Service Attacks.* Finally, we address a specific denial-of-service (DoS) attack against our scheme. This attack is possible because of the small packet size (29-byte payload in TinyOS [Crossbow Technology Inc. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm] used in sensor networks and, mutual authentication being the last step. Recall that the first message sent from an 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. Therefore, the MS has to send the message via multiple packets. A host node has to receive all the packets to reconstruct the message, compute the MS, and finally proceed 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 address these security weaknesses. First, a host node u and an MS execute the mutual authentication process before the MS provides the parameters regarding the task:

$$MS \rightarrow u : MS, seq, MAC(f(MS, u), MS \mid seq) \tag{8}$$

$$u \rightarrow MS : u, seq + 1, MAC(f(u, MS), u \mid seq + 1). \tag{9}$$

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 an MS or a regular sensor node. Recall that the ID for an 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 an MS will not fall into the interval $[1, 65536]$ because it is unlikely that all the other 6 bytes of an MS ID output from a hash function are all zeros. Nevertheless, in case that rare situation happens, we can change either T_s or T_e slightly to derive an MS ID falling outside of the interval $[1, N]$, without sacrificing much security. Thus, this approach can prevent a compromised sensor node from impersonating an 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. An 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 an 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, are 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.

3.3 Security and Performance Analysis

Next we analyze the security and the performance of our proposed scheme.

Security analysis. The security of our scheme is based on the assumption that the Merkle hash tree, the 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 [Rivest 1994]) 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 an MS only possesses one polynomial share, an attacker will not gain a further advantage from compromising an MS instead of a regular sensor node if its goal is to recover $f(x, y)$. Moreover, our scheme with the DoS-resistance enhancement enables a regular sensor node to verify every received packet from an 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 [Rivest 1994] 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 an MS node ID be 16 bits and 64 bits, respectively. For an 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 an 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 an MS node. It has been shown [Liu and Ning 2003b] 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 an 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 of these 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 1 byte [Ye et al. 2004]. Thus, the energy used by a host node to establish a pairwise key with an MS node is about the same as that used in transmitting 13 bytes. The computation cost of an 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 an 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: $(t + 1)$ coefficients. In addition to storing a polynomial share, an MS stores all the block IDs. As an example, suppose a sensor field is divided into 64 by 64 cells and the maximum size of a block is 8 cells, then we can use 2 bytes to represent a block ID. Hence, even if an MS is to traverse hundreds of blocks and its resources are as scarce as those of a MICA2-based mobile sensor [Bergbreiter and Pister 2003], the storage overhead is still not a concern.

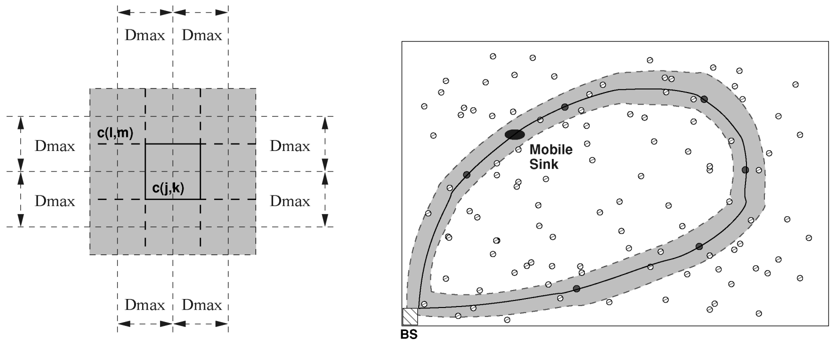
We will show some detailed results through our implementation in Section 5.

3.4 An Extension: Enabling Trajectory Changing

The trajectory discussed previously is fixed. However, to accomplish the mission, the MS may need to change its trajectory under certain conditions, such as meeting large obstacles (e.g., enormous rocks) or coverage holes (e.g., a big pool) along the original trajectory. In a battlefield scenario, such a condition could be the discovery of enemies' appearance in certain area of the trajectory. As another example, suppose the sensor network is deployed to monitor the temperature in an area and the MS is sent to collect the sensed data. When the MS collects an abnormal temperature datum, it needs to search around and find out how big the abnormal area is. Such investigation requires the MS to also change its trajectory.

To enable the MS to change its trajectory, we extend our scheme and propose a conditional privilege granting scheme. The basic idea is that, when certain conditions are met, the MS is allowed to expand its trajectory and access certain cells that are not on the original trajectory.

To achieve this goal, we integrate condition(s) into the ticket that is granted to the MS and we called this ticket a *conditional ticket*. Using the conditional ticket, the MS can access a strip of cells around the trajectory when certain conditions are satisfied. The strip of accessible cells can be determined by their distance to the trajectory. That is, if the distance of a cell from the trajectory is less than a system parameter D_{max}, this cell will be accessible to the MS. More formally.



(a) Determine whether a cell is conditionally accessible or not using system parameter D_{max}.

(b) Example of a strip of cells around the trajectory that is accessible to the MS when certain conditions are met.

Fig. 5. Illustration of conditional trajectory change.

Definition 3.1. Suppose node u is in $cell(l, m)$. Given D_{max}, condition C, and the last accessed cell $cell(j, k)$, if $| \; l - j \; | \leq D_{max}$ and $| \; m - k \; | \leq D_{max}$, we say that $cell(l, m)$ is *conditionally accessible* to the MS given condition C.

Figure 5(a) shows the conditional accessible cells around a cell on the trajectory. Figure 5(b) shows the strip of cells around the trajectory that are accessible to the MS using the Definition 1.

Next, we describe the conditional privilege-granting scheme in detail. In the original scheme, the MS only possesses one polynomial share and one ticket (the ID of the MS constructed using Equation (7). Similarly, we construct the conditional ticket as follows:

$$MS = H(TT \mid Ts \mid Te \mid X_{1m} \mid C_i), \tag{10}$$

where C_i is the condition(s) under which the MS is allowed to change the trajectory. C_i can be Arabic numbers such as $1, 2, 3, \ldots, n$, denoting various predefined conditions. Alternatively, it can be more descriptive, for example, $C_i = \{$the temperature is higher than 120F$^\circ\}$ or $C_i = \{$there are no sensors in cell(10,10)$\}$, and so on. Depending on the application, we can either preload D_{max} into each sensor if it is fixed, or add it into the conditional ticket if it is variable.

To establish a pairwise key with a node u in block B_i, the MS provides MS, TT, Ts, Te, and X_{1m}, together with several auxiliary values in the Merkle hash tree, which is similar to the original scheme. In addition, the MS also needs to provide C_i and $cell(j, k)$ to node u, where $cell(j, k)$ is the last successfully accessed cell on the trajectory.

When receiving these values from the MS, node u first checks whether C_i is in the ticket. If there is no C_i in the ticket, no trajectory change is needed, and node u and the MS can mutually authenticate each other as in the original scheme. Otherwise, if C_i is in the ticket and node u's cell is in the strip around the trajectory (based on D_{max}), node u will need to verify whether the condition (C_i) holds, before granting MS's request.

Next, we use a simple example to explain how to verify the condition C_i at a node u. Suppose condition C_i is equal to one, denoting trajectory changing is allowed when there is a coverage hole. If node u is close to the coverage hole, then, it can verify the existence of the hole in several ways. First, it may have observed the hole by itself, if it has never received any messages from the cells in the hole area for a long time. If this is the case, node u can confirm the condition at once and grant the request from the MS, if it can mutually authenticate with the MS based on the conditional ticket. Alternatively, when queried by the MS, node u can send a query message that contains C_i to its neighbors to check whether they have observed the same condition C_i as well. Note that this message should be authenticated by the pairwise key shared between node u and each of its neighbors, respectively. (Node u may also simply broadcast the message encrypted using the one-hop key, or cluster key, shared with its neighbors if the LEAP protocol [Zhu et al. 2003a] is used.) Each neighbor of u should generate a true/false report on condition C_i, attached, with a keyed MAC, and send it to node u. After collecting all the reports, if node u concludes that the condition holds, it proceeds to verify MS's conditional ticket and authorize MS's request.

In the spirit of this scheme, we can construct more complex conditional tickets or assign multiple conditional tickets to an MS. The security and performance overheads of regular sensors and mobile sensors are similar to that of the original scheme.

4. REVOKING A MOBILE SINK ON-DEMAND

If a mobile sink (MS) is detected to be 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. A simple approach would be for the the base station to 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.

4.1 The Basic Scheme

To multicast a revocation message within the revocation area, the basic idea of location-based multicasting [Ko and Vaidya 2000; Huang et al. 2003] can be applied. The base station first generates a message thaat indicates the ID of the MS to be revoked and the scope of the 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, if 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 3.2.2), this may need much 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 required 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.

4.2 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.

In the following, we first present two enhanced schemes, each of which is based on one of these 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 the messages. Instead, we can use the GPSR protocol [Bose et al. 1999; Karp and Kung 2000; Kuhn et al. 2003] 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 6(a) illustrates how to multicast revocation messages using the GPSR-based scheme. As opposed to the basic scheme where all the revocation packets follow the trajectory, in the GPSR-based scheme each packet is forwarded toward 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 illustrative purposes, 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.

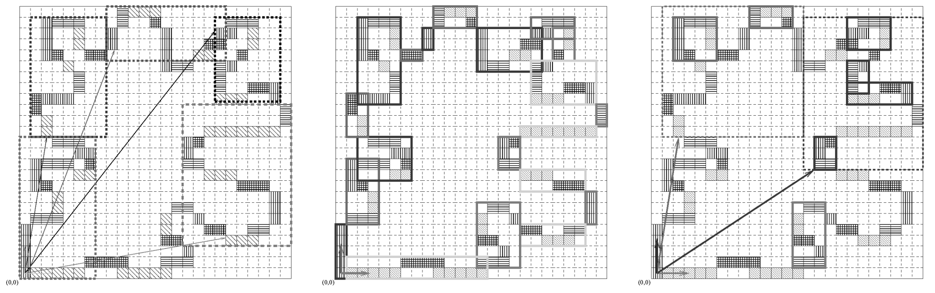

(a) OPT-I: GPSR-based scheme. (b) OPT-II: Minimum message scheme. (c) OPT-III: Optimized multi-message scheme.

Fig. 6. Illustration of the three enhanced privilege revocation schemes.

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 from the base station in the trajectory are always the last to receive the revocation notification. In practice, we should have high priority to notify these sensor nodes because it is likely a compromised MS has not contacted 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 toward a revocation area, all the nodes in the cells that 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 that cross the path along which the messages are 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 3.2.2, 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 programming-based approach to minimize the number.

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

$$B_1, B_2, \ldots, B_b.$$

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 $B_i, B_{i+1}, \ldots, B_b$ 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 are combined into at most k expanded blocks, denoted as $R_m(1,k)$, can be computed as follows:

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

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

A more detailed description of the algorithm is shown in Figure 7. Figure 6(b) 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. These 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—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.

In the following, we extend the algorithm described in Figure 7 to design a new algorithm that can minimize the overall communication cost. We consider the overall communication cost as 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 subscopes.

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.

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 $k$
                $R_m(i, l) = +\infty$
                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 $next(i, l)$
        $i = next(i, l)$
```

Fig. 7. The algorithm for the minimum message scheme.

—$R_m(i,l,e)$: 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 $B_i, B_{i+1}, \ldots, B_e$ 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.

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

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

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

Figure 6(c) 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.

4.3 Discussions

During the description of all four revocation schemes, we have implicitly assumed that a revocation packet can arrive at the first 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 [Perrig et al. 2001] for this purpose. Packet dropping cannot be prevented, but may be detected. The *watchdog* mechanism [Marti et al. 2000] and the reputation-based scheme [Ganeriwal and Srivastava 2004] may be employed to thwart a compromised node from dropping revocation messages.

4.4 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 is larger than 10. Four revocation algorithms will be evaluated: the basic algorithm, the GPSR-based scheme (OPT-I), the minimum message scheme (OPT-II), and the optimized multimessage scheme (OPT-III). All four of these schemes are 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 8, 9, 10, and 11, respectively.

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 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 a smaller number of blocks in each message. The smaller the number of blocks in a message, the faster

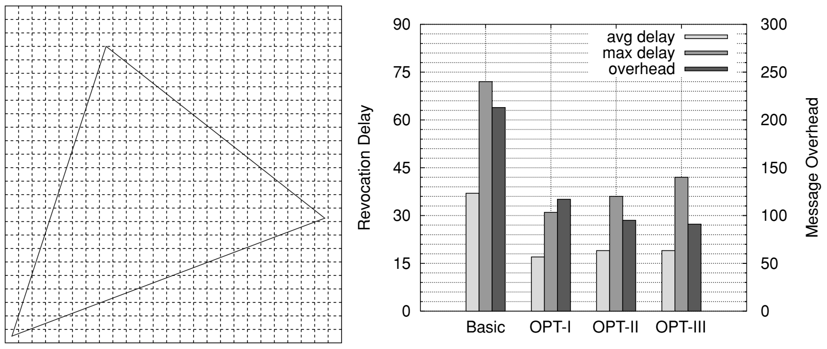


Fig. 8. Triangle trajectory.

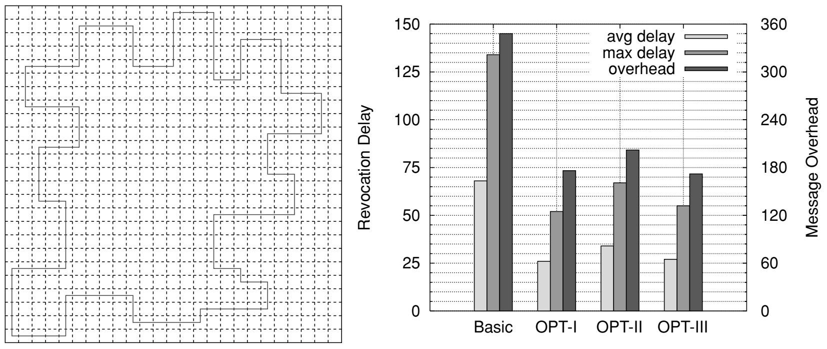


Fig. 9. Polygon trajectory.

the message can be forwarded. 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 outperform 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 9 and 11, 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.

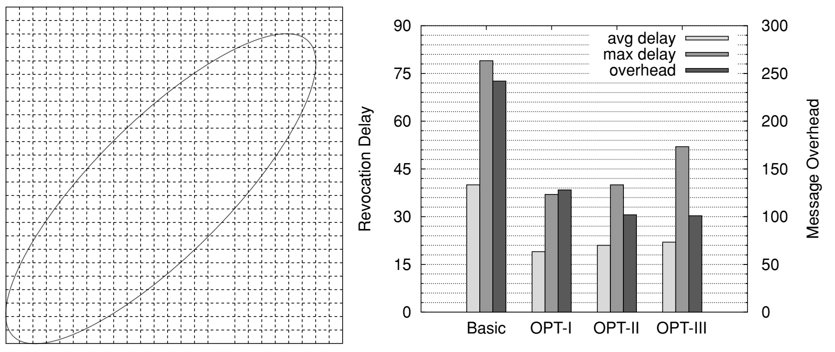


Fig. 10. Ellipse trajectory.

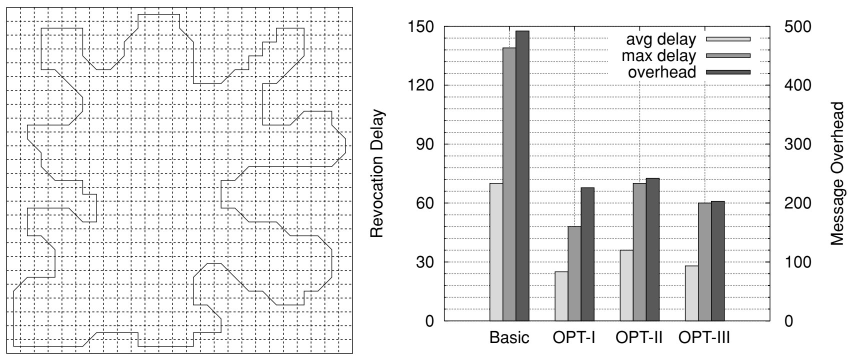


Fig. 11. Irregular shape trajectory.

5. IMPLEMENTATION

We have implemented our scheme of privilege restriction on MICA2 motes [Crossbow Technology Inc. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm]. in the TinyOS platform [Hill et al. 2000]. The programs were written in nesC [Gay et al. 2003], a C-like programming language used for developing applications in TinyOS. Our implementation does not include the privilege revocation schemes yet, since currently the basis of them, the GPSR scheme, is not implemented in TinyOS. We shall implement them when the GPSR scheme becomes available in TinyOS.

Our implementation is based on the tiny key management package (TinyKeyMan[1]) provided by Liu and Ning at North Carolina State University. TinyKeyMan provides an implementation of the polynomial pool-based key predistribution in sensor networks. In our case, we adapted it to assign the shares of the same polynomial to the sensors and the MS based on node ID. In addition, we added 2987 lines of new code in implementing our privilege restriction scheme.

[1]The TinyKeyMan package is available online at http://discovery.csc.ncsu.edu/software/TinyKeyMan.

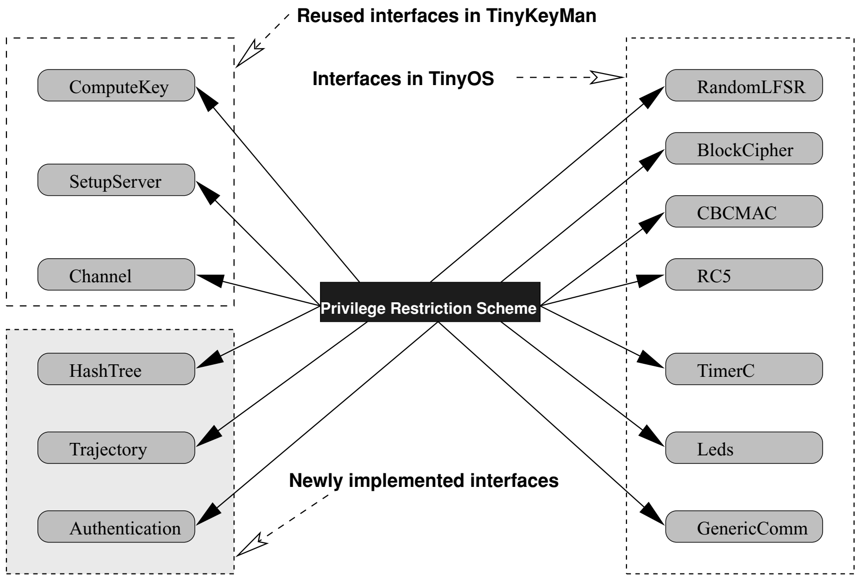


Fig. 12. The components and the interfaces used in implementing the proposed privilege restriction scheme. The seven interfaces on the right are directly adopted from TinyOS. On the left, the top four interfaces are provided in the TinyKeyMan package, and the bottom three are new interfaces implemented by us.

5.1 System Architecture

Figure 12 depicts the components and the interfaces used by our implementation. Of the interfaces shown in the figure, the seven interfaces on the right are directly adopted from TinyOS (with most of them from the TinySec package). For instance, we use the RC5 block cipher [Rivest 1994] to provide the CBC-MAC [Bellare et al. 2000].

The top four interfaces on the left are from the TinyKeyMan package. Among them, the *ComputerKey* interface is used for computing the pairwise key based on bivariate polynomial evaluation. The *SetupServer* interface assigns a polynomial share to a node based on its node ID. For the MS, the polynomial share is assigned based on its ID calculated using Equation 7. The *Channel* interface implements the channel module.

The bottom three interfaces on the left are interfaces implemented by us. The *Trajectory* interface implements the cell merging process; the *HashTree* interface implements the block compressing process; and the *authentication* interface implements the authentication procedure between a sensor node and the MS. The first two interfaces are implemented only in MS; whereas the *authentication* interfaces are implemented in both MS and static sensor nodes. In the following, we describe these interfaces and the packet formats in detail.

5.2 The Trajectory Interface

Currently, this interface includes four typical trajectories: triangle, polygon, ellipse, and irregular shape. An array that consists of the IDs of the cells on the trajectory is used to store one trajectory. For example, the ellipse trajectory shown in Figure 10 covers 79 cells. Each *cell ID* is represented by a dual pair (i, j), where i and j are the cell's x and y axis, respectively. Thus, in our implementation, the ellipse trajectory is represented by an array that contains 79 cell IDs. Given a trajectory, the cell merging algorithm shown in Figure 2 is called to merge the cells into blocks.

5.3 The HashTree Interface

Based on the set of blocks output from the *Trajectory* interface, a Merkle hash tree is constructed in the *HashTree* interface. We first blind the block IDs using the one-way hash function. Then, we compute the levels of the tree recursively from the leaf nodes to the root. A nonleaf node in the tree is a hash of its two child nodes, that is $X_i = H(X_i\text{'s left child} \mid X_i\text{'s right child})$, where H is a collision-resistant hash function. We considered three important issues when implementing the Merkle hash tree in TinyOS. (These issues are first discussed in Chapweske and Mohr [2002], where the Merkle hash tree is used for file integrity verification.)

First, the hash tree construction is only as strong as the underlying hash algorithm H. Thus, we use a secure hash algorithm CBC-MAC [Bellare et al. 2000] as the basis of the hash tree. CBC-MAC is efficient, and the fact that it relies on a block cipher minimizes the number of cryptographic primitives we must implement in the limited memory we have available. CBC-MAC is provably secure, however, the standard CBC-MAC construction is not secure for variable messages. Bellare et al. [2000] suggested three alternatives for generating MACs for variable sized messages. In our implementation, we adopted the CBC-MAC implementation in TinySec, which is the variant that XORs the encryption of the message with the first plaintext block.

Second, in order to reduce the collisions between leaf hashes and internal hashes, different hash constructions are used to hash the leaf nodes and the internal nodes. The same hash algorithm, CBC-MAC, is used as the basis of each construct, but a single 1 byte in network byte order, or 0x01 is appended to the input of the internal node hashes, and a single 0 byte, or 0x00 is appended to the input of the leaf node hashes.

Third, since a trajectory has an arbitrary number of blocks, the constructed Merkle hash tree might be unbalanced (the number of leaves is not a power of 2). In such a case, the tree is constructed as follows. Interim hash values of internal nodes that do not have a sibling value to which they may be concatenated are promoted, unchanged, up the tree until a sibling is found. For example, Figure 13 shows the hash tree for a trajectory made up of five blocks, B1, B2, B3, B4, and B5.

The *HashTree* interface can also generate the authentication path for a leaf node. The authentication path consists a set of auxiliary values in the Merkle hash tree that allows a sensor node in a certain leaf node, or block, to efficiently verify the MS's ID.

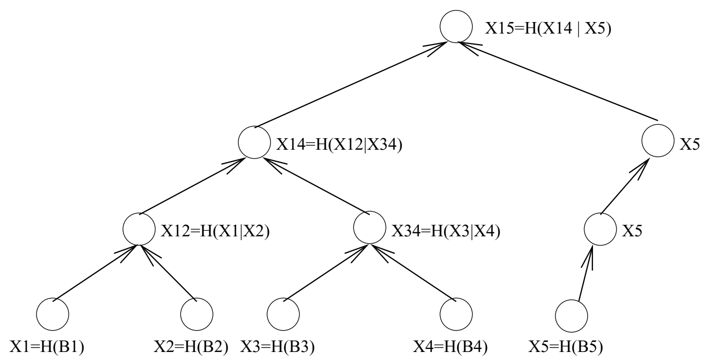


Fig. 13. Unbalanced Merkle hash tree. X_5 does not have any immediate siblings with which to be combined to calculate the next generation. So, X_5 is promoted up the tree, without being rehashed, until it can be paired with value X_{14}. The values X_5 and X_{14} are then concatenated, and hashed, to produce the root hash X_{15}.

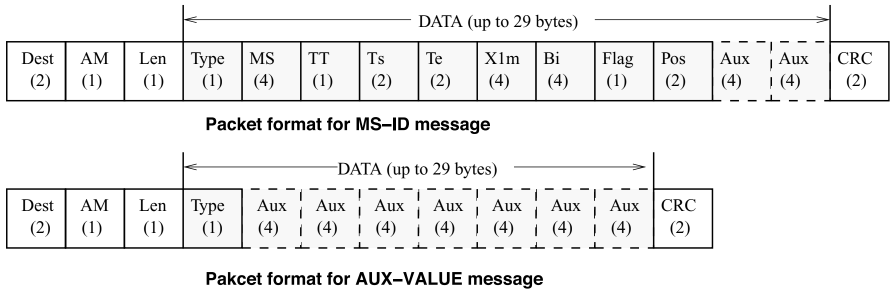


Fig. 14. Packet formats.

5.4 Packet Formats

Our implementation involves two new message types: MS-ID and AUX-Value. The packet formats for the two messages, shown in Figure 14, are based on the current packet format in TinyOS. The common fields are *destination address, active message (AM) type*, and *length*. Active message types are similar to port numbers in TCP/IP. The AM type specifies the appropriate handler function to extract and interpret the message on the receiver. The data field is up to 29 bytes in a TinyOS packet. In these two packets, the *type* field designates the message type, which is 7 for the first and 8 for the second. Type 1–6 Messages are previously defined in the TinyKeyMan package.

5.4.1 *MS-ID Message Packet Format.* An MS-ID message is used to send MS, TT, T_s, T_e, X_{1m}, B_i, and flag, the auxiliary values' position information, and up to two auxiliary values (optional) on the authentication path to the sensor.

The auxiliary values' position information is needed for a sensor to correctly reconstruct X_{1m}, the root of the Merkle hash tree, using the authentication path. Recall that a nonleaf node is the hash over the concatenation of its left and right children, where the left child is before the right child. We will get a different hash if the right child is before the left child. Thus, the sensor must have

X X	Y	Z Z Z Z Z

Fig. 15. The flag field in the MS-ID message.

the position (left or right) information of each auxiliary value; otherwise it cannot reconstruct X_{1m} successfully even if it has the correct authentication path.

The auxiliary values' position information is contained in the *Pos* field in an MS-ID message. Each of its bits represents the position of one auxiliary value. If the corresponding auxiliary should be on the left, the bit is set to 0; otherwise the bit is set to 1. Since *Pos* field is 2 bytes, it can be used to hold up to 16 auxiliary values. This means that the Merkle hash tree could have up to 64, 000 blocks, which should be big enough to store an enormous trajectory.

The *flag* field, shown in Figure 15, is described as follows. The first 2 bits (*XX*) show how many auxiliary values are behind. *XX* could be 0, 1, or 2, which means that zero, one or two auxiliary values are behind, separately. The last 5 bits (*ZZZZZ*) designate the total number of auxiliary values on the authenticate path. The third bit (Y) indicates whether all the auxiliary values are in the MS-ID message or not. If the total number of auxiliary values is not larger than two, $Y = 0$; otherwise $Y = 1$, which indicates that this message will be followed by an AUX-VALUE message.

5.4.2 *AUX-VALUE Message Packet Format.*

An AUX-VALUE message is used for sending auxiliary values on the authentication path. Each AUX-VALUE message can hold up to seven auxiliary values, since each auxiliary value is 4 bytes. Combining the MS-ID message with one AUX-VALUE message, we can send nine auxiliary values using two messages. Thus, by piggybacking two auxiliary values in an MS-ID message, we can support the authentication of a trajectory of up to 512 blocks ($2^9 = 512$), which is often big enough to hold a typical trajectory.

5.5 The Authentication Interface

Lastl, the *Authentication* interface implements the message sending and receiving in the authentication procedure between a sensor node and MS. Its implementation on MS is different from that of on a static sensor node.

On MS, before sending any messages, the Authentication interface first calls the Trajectory interface to merge cells into blocks; then the HashTree interface is called to generate X_{1m}, which, in turn, is used to generate mobile sink's ID *MS*. After that, MS sends an MS-ID message and an AUX-VALUE message to the current senor node u on the trajectory, using the Channel interface. If node u can successfully verify the MS ID, similar to the strawman scheme in Section 3.1, MS sends an authentication message (Equation 2) to node u. Later, when MS receives the reply message (Equation 3) from node u, it can confirm whether node u has the same pairwise key by verifying the MAC in the reply message.

On a static sensor node, the Authentication interface processes the received MS-ID message and AUX-VALUE message to verify the ID of MS and compute

Table I. The Required ROM/RAM Space for MS with Different Trajectories

Trajectory	Triangle	Polygon	Ellipse	Irregular Shape
ROM (bytes)	24,792	24,806	24,916	24,926
RAM (bytes)	609	597	657	771

the pairwise key shared with MS by evaluating its polynomial share on MS's ID. After it receives an authentication message (Equation 2) from the MS, it can check whether it has the same pairwise key as MS by verifying the MAC in the message. If it can successfully verify the MAC, the message (Equation 3) will be sent back to the MS.

5.6 TinyOS Code Size

We upload and run the code in MICA2 motes. Since the TinyOS code of an MS is different from that of a static sensor node, MS and static sensor nodes have different code sizes. In the case of MS, the memory used for code and data depends on the trajectory. A more complex trajectory often consists of a greater number of blocks. Since this block information needs to be stored in the MS, a more complex trajectory often requires more ROM/RAM space. This situation can be observed from Table I, which shows the usage of ROM/RAM for MS with four different trajectories. The four trajectories are shown in Figures 8, 9, 10, and 11. For the most complex trajectory, the irregular shape, the required code space is 25 KB (out of 128 KB) in ROM and about 800 Bytes (out of 4 KB) in RAM for the MS.

For the static sensor node, the required code space is 17.2 KB (out of 128 KB) in ROM and 672 Bytes (out of 4 KB) in RAM. From these results, we can see that the memory requirements for the privilege restriction scheme are reasonable given that MICA2 motes have 4 KB of RAM and 128 KB of ROM for data and code storage.

6. 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) [Perrig et al. 2001], schemes exploiting the initial trustworthiness of newly deployed sensors [Zhu et al. 2003a], and schemes based on the framework of probabilistic key predeployment [Chan et al. 2003; Du et al. 2003; Eschenauer and Gligor 2002; Liu and Ning 2003b; Zhu et al. 2003b; Chan and Perrig 2005]. The Blundo scheme [Blundo et al. 1993], 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 [Du et al. 2003; Liu and Ning 2003b; Huang et al. 2004]. Our schemes can also be constructed from these extended schemes if necessary, although we used the Blundo scheme.

Perrig et al. [2001] presented μTESLA for base station broadcast authentication, based on one-way key chains and delayed key disclosure. Liu and Ning

[2003a] proposed several multilevel μTESLA schemes to further extend the scalability of the original μTESLA. Zhu et al. [2003a] and Deng et al. [2003] 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. [2004] and Ye et al. [2004] 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 an MS and a host node should be confidential. Moreover, these schemes cannot restrict the privilege of an MS if employed directly.

Wood and Stankovic [2002] identified a number of DoS attacks in sensor networks. Deng et al. [2004] 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 [2003] described several security attacks on routing protocols for sensor networks. The attacks that are difficult to detect or prevent are the ones that combine *sinkhole* and *wormhole* attacks [Hu et al. 2003]. These attacks however do not directly apply to our schemes. Zhang and Cao [2005] proposed predistribution 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 [Ko and Vaidya 2000] and sensor networks [Huang et al. 2003]. 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 take a great deal of 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 accurately predetermined, while the revocation area in our schemes can be prespecified, which allows the optimizations proposed in this article.

7. CONCLUSIONS AND FUTURE WORK

In this article, we identified new security challenges in using MSs; if an MS is given too many privileges, the compromise of the MS may break down the whole network; on the other hand, without some necessary privileges, 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 privileges of the MS without impeding its capability to carry out any authorized operations for an assigned task. In addition, we present an extension to allow conditional trajectory change due to unexpected events. To further reduce the possible damage caused by a compromised MS, we then proposed efficient message forwarding schemes for depriving the privileges assigned to the compromised MS immediately after its compromise has been detected. Detailed analysis, simulation, and implementation 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 article to address the MS privilege issues in wireless sensor networks. As the initial work, we do not expect to solve all the problems. In our future work we will consider the following two issues. First, we will support policies more flexible than the trajectory-based one. For example, both space and time may have different degrees of granularity and logical operations such as OR and AND may be needed. Second, in a sensor application, multiple mobile sinks may be dispatched for either the same or different tasks with overlapped time periods or trajectories. We will examine its impact on both security and performance, and see if some level of performance optimization is possible.

REFERENCES

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. 2002. Wireless sensor networks: A survey. *Comput. Netw. 38,* 4.

Anjum, F., Pandey, S., Kim, B., and Agrawal, P. 2005. Secure localization in sensor networks using transmission range variation. In *Proceedings of the IEEE Conference on Mass Storage Systems and Technologies (MASS)*. 195–203.

Bellare, M., Kilian, J., and Rogaway, P. 2000. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci. 61,* 3, 362–399.

Bergbreiter, S. and Pister, K. 2003. Cotsbots: An off-the-shelf platform for distributed robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Roloats and Systems (IROS'03)*.

Blundo, C., Santis, A. D., Herzberg, A., Kutten, S., Vaccaro, U., and Yung, M. 1993. Perfectly-secure key distribution for dynamic conferences. In *Proceedings of Advances in Cryptology, (CRYPTO'92)*. Lecture Notes in Computer Science, vol. 740. 471–486.

Bose, P., Morin, P., Stojmenovic, I., and Urrutia, J. 1999. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM '99)*. ACM Press, New York, NY, 48–55.

Capkun, S., Cagalj, M., and Srivastava, M. 2006. Securing localization with hidden and mobile base stations. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '06)*. Barcelona, Spain.

Capkun, S. and Hubaux, J. 2002. Secure positioning in sensor networks. *Tech. Rep.* EPFL/IC/200444 available at http://www.terminodes.org/micsPublications.php. 1278–1287.

Chan, H. and Perrig, A. 2005. Pike: Peer intermediaries for key establishment in sensor networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*.

Chan, H., Perrig, A., and Song, D. 2003. Random key predistribution schemes for sensor networks. In *Proceedings of the IEEE Security and Privacy Symposim 2003*.

Chapweske, J. and Mohr, G. 2002. Tree hash exchange format (thex). http://open-content.net/specs/draft-jchapweske-thex-01.html.

Crossbow Technology Inc. Wireless sensor networks. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

Deng, J., Han, R., and Mishra, S. 2003. Security support for in-network processing in wireless sensor networks. In *Proceedings of 1st ACM Workshop on the Security of Ad Hoc and Sensor Networks (SASN'03)*.

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

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

Eschenauer, L. and Gligor, V. 2002. A key-management scheme for distributed sensor networks. *Proceedings of the ACM Computer and Communications Security Conference (CCS'02)*.

Ganeriwal, S., Capkun, S., Han, C.-C., and Srivastava, M. B. 2005. Secure time synchronization service for sensor networks. In *Proceedings of the 4th ACM Workshop on Wireless Security (WiSe'05)*. ACM Press, New York, NY, 97–106.

Ganeriwal, S., Kumar, R., and Srivastava, M. 2003. Timing-sync protocol for sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys'03)*.

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

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., and Culler, D. 2003. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03)*. ACM Press, New York, NY, 1–11.

Goldreich, O., Goldwasser, S., and Micali, S. 1986. How to construct random functions. *J. ACM 33,* 4, 210–217.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. 2000. System architecture directions for networked sensors. In *Proceedings of the 9th Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*.

Hu, Y., Perrig, A., and Johnson, D. 2003. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. *Proceedings of the ACM Computer on Communications Security Conference (INFOCOM'03)*.

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

Huang, Q., Lu, C., and Roman, G. 2003. Spatiotemporal multicast in sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (Sensys'03)*.

Kansal, A., Somasundara, A. A., Jea, D. D., Srivastava, M. B., and Estrin, D. 2004. Intelligent fluid infrastructure for embedded networks. In *Proceedings of the International Conference on Mobile Systems Applications and Services (MobiSys'04)*. 111–124.

Karlof, C. and Wagner, D. 2003. Secure routing in sensor networks: attacks and countermeasures. In *Proceedings of the First IEEE Workshop on Sensor Network Protocols and Applications*.

Karp, B. and Kung, H. 2000. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Aunual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom'00)*.

Ko, Y. and Vaidya, N. 2000. Geotora: A protocol for geocasting in mobile ad hoc networks. *Proceedings of the International Conference on Network Protocols (ICNP)*.

Kuhn, F., Wattenhofer, R., and Zollinger, A. 2003. Worst-case optimal and average-case efficient geometric ad hoc routing. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'03)*. ACM Press, New York, NY, 267–278.

Lazos, L. and Poovendran, R. 2005. Serloc: Robust localization for wireless sensor networks. *ACM Trans. Sen. Netw. 1,* 1, 73–100.

Liu, D. and Ning, P. 2003a. 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)*. 263–276.

LIU, D. AND NING, P. 2003b. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*. 52–61.

LIU, D., NING, P., AND DU, W. 2005. Attack-resistant location estimation in sensor networks. In *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*.

MARTI, S., GIULI, T., LAI, K., AND BAKER, M. 2000. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the Annual Conference on Mobile Computing and Networking (MobiCom'00)*.

MCMICKELL, M. B., GOODWINE, B., AND MONTESTRUQUE, L. A. 2003. Micabot: A robotic platform for large-scale distributed robotics. In *Proceedings of the IEEE International Conference on Robtics & Automation*.

MERKLE, R. 1989. A certified digital signature. In *Proceedings of Advances in Cryptography*. 218–238.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, J. 2001. Spins: Security protocols for sensor networks. In *Proceedings of the Annual Conference on Mobile Computing and Networking (MobiCom'01)*.

PRIYANTHA, N. B., CHAKRABORTY, A., AND BALAKRISHNAN, H. 2000. The cricket location-support system. In *Proceedings of the Annual Conference on Mobile Computing and Networking (MobiCom)*. 32–43.

PRIYANTHA, N. B., MIU, A. K., BALAKRISHNAN, H., AND TELLER, S. 2001. The cricket compass for context-aware mobile applications. In *Proceedings of the Annual Conference on Mobile Computing and Networking (MobiCom)*.

RIVEST, R. 1994. The rc5 encryption algorithm. In *Proceedings of the 1st International Workshop on Fast Software Encryption*. 86–96.

SALTZER, J. H. AND SCHROEDER, M. D. 1975. The protection of information in computing systems. *Proc. IEEE*.

SAVVIDES, A., HAN, C., AND SRIVASTAVA, M. 2001. Dynamic fine-grained localization in ad hoc networks of sensors. In *Proceedings of the Annual Conference on Mobile Computing and Networking (MobiCom)*. 166–179.

SIBLEY, G., RAHIMI, M., AND SUKHATME, G. 2002. Robomote: A tiny mobile robot platform for large-scale ad hoc sensor networks. In *Proceedings of the IEEE International Conference on Robtics & Automation*. Vol. 2. Washington D.C., 1143–1148.

SONG, H., ZHU, S., AND CAO, G. 2007. Attack-resilient time synchronization for wireless sensor networks. *Ad Hoc Netw. 5,* 1 (Jan.), 112–125.

SUN, K., NING, P., AND WANG, C. 2006. Secure and resilient clock synchronization in wireless sensor networks. *IEEE J. Sel. Areas Commun. 24,* 2 (Feb.), 395–408.

TIRTA, Y., LI, Z., LU, Y., AND BAGCHI, S. 2004. Efficient collection of sensor data in remote fields using mobile collectors. In *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN'04)*.

WOOD, A. AND STANKOVIC, J. 2002. Denial of service in sensor networks. *IEEE Comput.*, *35*, 10, 54–62.

XU, Y., HEIDEMANN, J., AND ESTRIN, D. 2001. Geography informed energy conservation for ad hoc routing. In *Proceedings of the Annual Conference on Mobile Computing and Networking (MOBICOM'01)*.

YE, F., LUO, H., CHENG, J., LU, S., AND ZHANG, L. 2002. A two-tier data dissemination model for large-scale wireless sensor networks. In *Proceedings of the Annual Conference on Mobile Computing and Networking (MOBICOM'02)*, 148–159.

YE, F., LUO, H., LU, S., AND ZHANG, L. 2004. Statistical en route filtering of injected false data in sensor networks. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*.

ZHANG, W. AND CAO, G. 2004. Dctc: Dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Trans. Wirel. Commun. 3,* 5 (Sept.), 1689–1701.

ZHANG, W. AND CAO, G. 2005. Group rekeying for filtering false data in sensor networks: A predistribution and local collaboration based approach. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*.

Zhang, W., Cao, G., and LaPorta, T. F. 2007. Data dissemination with ring-based index for wireless sensor networks. *IEEE Trans. Mobile Comput. 6,* 7, 832–847.

Zhu, S., Setia, S., and Jajodia, S. 2003a. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*. ACM Press, New York, NY, 62–72.

Zhu, S., Setia, S., Jajodia, S., and Ning, P. 2004. An interleaved hop-by-hop authentication scheme for filtering false data in sensor networks. *Proceedings of the IEEE Symposium on Security and Privacy*.

Zhu, S., Xu, S., Setia, S., and Jajodia, S. 2003b. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. *Proceedings of the IEEE International Conference on Network Protocol (ICNP)*.

Received January 2006; revised August 2006, May 2007; accepted December 2007