

p DCS: Security and Privacy Support for Data-Centric Sensor Networks

Min Shao[†], Sencun Zhu[†], Wensheng Zhang[‡], and Guohong Cao[†]

[†] Department of Computer Science & Engineering
The Pennsylvania State University
Email: {mshao,szhu,gcao}@cse.psu.edu

[‡] Department of Computer Science
Iowa State University
Email: wzhang@cs.iastate.edu

Abstract—The demand for efficient data dissemination/access techniques to find the relevant data from within a sensor network has led to the development of data-centric sensor networks (DCS), where the sensor data as contrast to sensor nodes are named based on attributes such as event type or geographic location. However, saving data inside a network also creates security problems due to the lack of tamper-resistance of the sensor nodes and the unattended nature of the sensor network. For example, an attacker may simply locate and compromise the node storing the event of his interest. To address these security problems, we present p DCS, a privacy-enhanced DCS network which offers different levels of data privacy based on different cryptographic keys. In addition, we propose several query optimization techniques based on Euclidean Steiner Tree and Keyed Bloom Filter to minimize the query overhead while providing certain query privacy. Finally, detailed analysis and simulations show that the Keyed Bloom Filter scheme can significantly reduce the message overhead with the same level of query delay and maintain a very high level of query privacy.

Index Terms: Security, privacy, data-centric, keyed bloom filter, wireless sensor networks,

I. INTRODUCTION

As sensor networks scale in size, so will the amount of sensing data generated. The large volume of data coupled with the fact that the data are spread across the entire network creates a demand for efficient data dissemination/access techniques to find the relevant data from within the network. This demand has led to the development of data centric sensor networks (DCS) [1], [2], [3], [4].

DCS exploits the notion that the nature of the data is more important than the identities of the nodes that collect the data. Thus, sensor data as contrasted to sensor nodes are “named”, based on attributes such as event-type (e.g., elephant-sightings) or geographic location. According to their names, the sensing data are passed to and stored at corresponding sensor nodes determined by a mapping function such as Geographic Hash Table (GHT) [1]. As the sensing data with the same name are stored in the same location, queries for data of a particular name can be sent directly to the storing nodes using geographic routing protocols such as GPSR [5], rather than flooding the query throughout the network. Figure 1 shows an example of using a DCS-based sensor network to monitor the activities or presence of animals in a wild animal habitat. The sensed data can be used by zoologists to study the animals, and may also be used to assist an authorized hunter to locate certain types of animals (e.g., boars and deers) for hunting. With DCS, all the sensing data regarding one type of animals are forwarded to and stored in one location. As a result, a zoologist only needs to send one query to the right location to find out the information about that type of animals. Similarly, a soldier can easily obtain enemy tank information through a DCS-based sensor network in the battlefield.

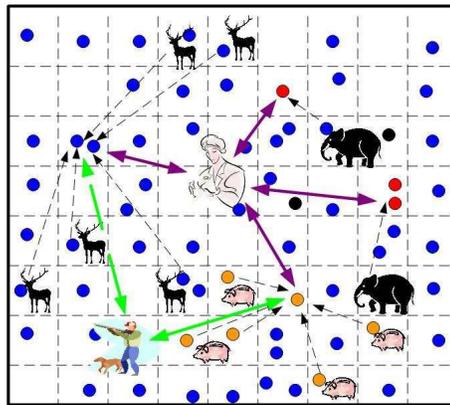


Fig. 1. A DCS-based sensor network which can be used by zoologists (who are authorized to know the locations of all animals) and hunters (who should only know the locations of boars and deers, but not elephants).

In many cases, DCS-based data dissemination offers a significant advantage over previous external storage-based data dissemination approaches, where an external base station (*BS*) is used for collecting and storing the sensing data. If many queries are issued from nodes within the network [4], external storage-based scheme is very inefficient since data must be sent back and forth between the sensors and the *BS*, thus causing the nodes close to the *BS* to die rapidly due to energy depletion. Further, for sensor networks deployed in hostile environments such as a battlefield, external *BS* may not be available because the *BS* is very attractive for physical destroy and compromise.

The previous DCS systems however were not designed with security in mind. All data of the same event type are stored at the same node [1] or several nodes [4] based on a publicly-known mapping function. As long as the mapping function and the types of events monitored in the system are known, one can easily determine the locations of the sensors storing different types of data. In our previous example, a zoologist can use the DCS system to locate any animals of interest. A non-authorized person may also use the DCS system to discover the locations of the animals for hunting. However, we may only permit a hunter to hunt some animals (e.g., boars and deers) but not the protected ones (e.g., elephants). A non-conforming hunter however may acquire the locations of the protected animals for hunting purpose. As such, security and privacy should be provided for DCS system.

In this paper, we present p DCS, a privacy enhanced DCS system for wireless sensor networks. To the best of our knowl-

edge, $pDCS$ is the first one to provide security and privacy to data-centric sensor networks. Specifically, $pDCS$ offers different levels of data privacy based on different cryptographic keys. Even if an attacker can compromise a sensor node and obtain all its keys, he cannot decrypt the data stored in the compromised node. In addition, we propose several query optimization techniques to minimize the query overhead, and use simulations to demonstrate that the proposed techniques can significantly reduce the message overhead without losing any query privacy.

The rest of the paper is organized as follows. In the next section, we discuss the assumptions and design goals. Section III presents several secure mapping functions, followed by optimization techniques for sending queries. In Section IV, we compare the performance of several query methods. The related work is discussed in Section V. Section VI concludes the paper.

II. ASSUMPTIONS AND DESIGN GOALS

As in other DCS systems [1], [4], our $pDCS$ system also assumes that a sensor network is divided into cells (or grids) where each pair of nodes in neighboring cells can communicate directly with each other. Cell is the minimum unit for detecting events (referred to as *detection cell*) and for storing sensor data (referred to as *storage cell*). A cell head coordinates all the actions inside a cell. Each cell has a unique id and every sensor node knows in which cell it is located. We also assume the clocks of sensor nodes in a network are loosely synchronized.

A. Attack Model

Given the unattended nature of a sensor network, an attacker may launch various security attacks against the network at all layers of the protocol stack [6]. Due to the lack of an one-for-all solution, in the literature these attacks are studied separately and the proposed defense techniques are also attack-specific. As such, we will focus on the specific security problems in our $pDCS$ network instead of solving all attacks. We assume that in a $pDCS$ network the (ultimate) goal of an attacker is to obtain the event data of his interest. To achieve this goal, an attacker may launch the following attacks.

- **Passive Attack** An attacker may passively eavesdrop on the message transmissions in the network.
- **Query Attack** An attacker may simply send a query into the network to obtain the sensor data of interest to him.
- **Readout Attack** An attacker may capture some sensor nodes and read out the stored sensor data directly. It is not hard to download data from both the RAM and ROM spaces of sensor nodes [7].
- **Mapping Attack** In this attack, the goal of an attacker is to identify the mapping relation between two cells. Specifically, he may either identify the storage cell for a specific detection cell or figure out the detection cell for a storage cell of his interest. Mapping attack is normally followed by a readout attack.

The passive attack can be relatively easily addressed by message encryption with keys of sufficient length, and the query attack can be addressed by source authentication [8] so that a node only answers queries from authorized entity. Given that compromising nodes is much easier than breaking the underlying encryption/authentication algorithm, we believe the readout attack and the mapping attack are more preferable to the attacker.

Note that letting detection cells encrypt sensor data and store the encrypted data *locally* cannot address the readout attack because an attacker can read out the encryption keys from the captured sensor nodes as well.

B. Security Assumptions

We assume that only nodes in a small number (s) of *cells* can be compromised. Although technically an attacker can compromise an arbitrary number of current generation of sensor nodes without much effort. In practice, it is difficult. For example, it may not be easy for sensor nodes to be captured because of their locations. Also, the attacker needs to spend longer time on compromising more sensor nodes, which may increase the chance of being identified. For simplicity, we say a cell is compromised when at least one node in the cell is compromised. To deal with the worst scenario, we allow an attacker to *selectively* compromise s cells.

Given the rich literature in key management, we do not address key management issues in this paper. We assume two nodes can establish a pairwise key based on one of many schemes [9], [10], [11], [12], and nodes in the same cell can establish a cell key using the simple hashing-based location-binding keying scheme [13]. Further, a group-wide shared keys may be established using our previous techniques developed in [11], [14].

We assume the existence of anti-traffic analysis techniques if so required. If an attacker is capable of monitoring and collecting all the traffic in the network, he may be able to correlate the detection cells and the storage cells without knowing the mapping functions. Therefore, we assume one of the existing schemes [15], [16], [17] may be applied to counter traffic analysis if the attacker is assumed to be capable of traffic analysis.

C. Design Goal

Our goal is to address the types of attacks that are specific to $pDCS$, i.e., passive attack, query attack, readout attack, and mapping attack. Since passive attack and query attack are easy to address, below we mainly discuss the requirements to be met for addressing the readout attack and the mapping attack.

- **Event Data Confidentiality** Even if an attacker can compromise a sensor node and obtain all its keys, he should be prevented from knowing the event data stored in the compromised node.
- **Backward Event Privacy** An attacker should be prevented from obtaining the previous sensor data for an event of his interest even if he has compromised some nodes.
- **Forward Event Privacy** We should also thwart (if not completely prevent) an attacker from obtaining the sensor data regarding an event in the future even if he has compromised some nodes.
- **Query Efficiency** Although security is not free, the scheme should not be too costly for sensor networks. Especially, it should be convenient and efficient for a legitimate Mobile Sink (MS) (i.e., mobile sensor, user or soldier) to issue his query without relying on network-wide flooding.
- **Query Privacy** A MS query should reveal as little location information of the sensor data as possible. For example, if multiple events are mapped and stored in the same storage cell, a

query for one of the events will also reveal the storage cell of the other events. As such, an attacker may eavesdrop on MS queries to minimize his efforts in launching a mapping attack.

III. *p*DCS: PRIVACY ENHANCED DATA-CENTRIC SENSOR NETWORKS

In this section, we first give an overview of the operations in *p*DCS. Then we present several schemes to randomize the mapping function. Finally, we describe optimization techniques for issuing queries.

A. The Overview of *p*DCS

Our solution involves five basic steps in handling sensed data: determine the storage cell, encrypt, forward, store, and query. We demonstrate the whole process through an example in which a cell u detects an event E .

1. Cell u first determines the location of the storage cell v through a keyed hash function.
2. u encrypts the recorded information (M_e) with its cell key. To enable MS queries, either the event type E or the detection time interval T is in its plain text format, subject to the requirement of the application.
3. u then forwards the message towards the destination storage cell. Here, techniques [15] should be applied to prevent traffic analysis and to prevent an attacker from injecting false packets.
4. On receiving the message, v stores it locally.
5. If an authorized mobile sink (MS) is interested in the event E occurred in cell u , it determines the storage cell v and sends a query there (optimized query schemes will be discussed in Section III-C).

The first step is to defend against the mapping attack. Without the mapping key, an attacker cannot determine the mapping from the detection cell to the storage cell. The second step is for preventing the readout attack. Since the storage cell v does not possess the decryption key for M_e , an attacker is prevented from deciphering M_e after he has compromised a node in v . Step 3 and Step 4 deal with forwarding and storing the sensing data, Step 5 shows the basic operation for issuing a MS query.

The following subsections focus on the performance and security issues related to Step 1, Step 2, and Step 5. Currently we assume some existing schemes [15], [4] for Step 3 and Step 4; we believe research in these areas bears its own importance and deserves independent study.

B. Privacy Enhanced Data-Location Mapping

From the system overview, we can see that an attacker can launch various attacks if he can find the correct mapping relation between a detection cell and a storage cell. This motivates our design of secure mapping to randomize the mapping relation among cells. Below we present three representative secure mapping schemes in the order of increasing privacy. The following notations are used during the discussion. Let N be the number of cells in the field, N_r and N_c be the number of rows and the number of columns, respectively. Every cell is uniquely identified with $L(i, j)$, $0 \leq i \leq N_r - 1$ and $0 \leq j \leq N_c - 1$.

To quantify and compare the privacy levels of different schemes, we assume that an attacker is capable of compromising totally s cells of his choice. To simplify the analysis, we assume

that there are m detection cells for the event of interest to the attacker, and the locations of these m cells are independent and identically distributed (iid) over N cells (In real applications, the locations of these m detection cells may correlate). We further introduce the concept of *event privacy level*.

Definition 1: Event Privacy Level (EPL) is the probability that an attacker *cannot* obtain both the sensor data and the encryption keys for an event of his interest.

According to this definition, the larger the EPL, the higher the privacy. This definition can be easily extended to the concepts of backward event privacy level (BEPL) and forward event privacy level (FEPL).

B.1 Scheme I: Group-key-based Mapping

In this scheme, all nodes store the same type of event E in the same location (L_r, L_c) based on a group-wide shared key K . Here

$$L_r = H(0|K|E) \text{ Mod}(N_r), L_c = H(1|K|E) \text{ Mod}(N_c) \quad (1)$$

To prevent the stand-alone readout attack, a cell should not store its data in its own cell. Hence, if a cell (x, y) finds out its storage cell is the same, i.e., $L_r = x$ and $L_c = y$, it applies H on L_r and L_c until either $L_r \neq x$ or $L_c \neq y$. To simplify the presentation, however, we will not mention this special case again during the following discussions.

Type I Query: A MS can answer the following query with one message: *what is the information about an event E ?* This is because all the information about event E is stored in one location. A MS first determines the location based on the key K and E , then sends a query to it directly to fetch the data using for example the GPSR protocol [5] (we will discuss several query methods with optimized performance and higher query privacy shortly).

Security and Performance Analysis: In this scheme, all m detection cells are mapped to one storage cell. An attacker first randomly compromises a node to read out the group key, based on which he locates the storage cell for the event. Because the data stored in the compromised node are encrypted by individual cell keys and the detection cell ids are encrypted as well, the attacker has to randomly guess the m detection cells. Assume that an attacker can compromise up to s cells. If the first compromised cell is the storage cell¹ (with probability $1/N$), the attacker will randomly compromise $(s - 1)$ cells from the rest $(N - 1)$ cells. There are totally $\binom{N-1}{s-1}$ combinations, among which $\binom{N-1-m}{s-1-i} \binom{m}{i}$ combinations correspond to the case where i out of m detection cells are all compromised. On the other hand, in the case when the first compromised node is not the storage cell (with probability $(N - 1)/N$), the attacker first compromise the storage cell, then randomly compromise $(s - 2)$ cells from the rest $(N - 2)$ cells. There are totally $\binom{N-2}{s-2}$ combinations, among which $\binom{N-2-m}{s-2-i} \binom{m}{i}$ combinations correspond to the case where i out of m detection cells are all compromised. Also note that an attacker can only obtain $\frac{i}{m}$ of the

¹For simplicity, we ignore the case when the first compromised cell is a detection cell. Our study shows that the error introduced by this simplification is negligible.

event data when i out of m detection cells are compromised. Let $B_1 = \min(s-1, m)$ and $B_2 = \min(s-2, m)$, then the BEPL of this scheme is

$$p_b^1(m, s) = 1 - \frac{1}{N} \sum_{i=1}^{B_1} \binom{i}{m} \binom{N-1-m}{s-1-i} \binom{m}{i} / \binom{N-1}{s-1} \\ - \frac{N-1}{N} \sum_{i=1}^{B_2} \binom{i}{m} \binom{N-2-m}{s-2-i} \binom{m}{i} / \binom{N-2}{s-2}$$

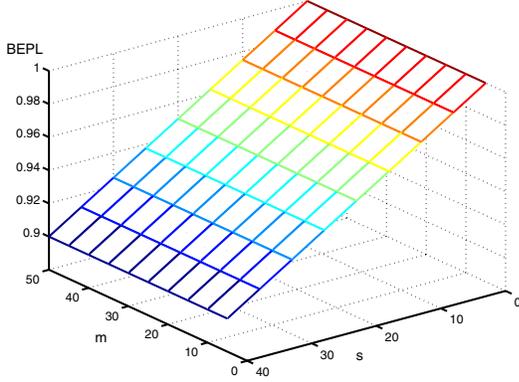


Fig. 2. The BEPL as a function of m and s , where m is the number of detection cells and s the number of compromised cells

Figure 2 shows the analytical result of BEPL as a function of m and s for a network size of $N = 20 * 20 = 400$ cells, from which we can make two observations. First, without surprise, BEPL decreases with s . Second, BEPL does not change with m . This is due to the tradeoff between the number of detection cells and storage cells that are probably compromised and the fraction of event data possessed by the compromised storage cells.

Suppose the attacker compromises s cells including the storage cell at time t_0 . He takes over these cells and can come back at a time t_1 in the future to obtain the event data from the storage cell and then simply decrypt all the data that are detected by these s cells during t_0 and t_1 . Assume that m cells will detect the event during t_0 and t_1 and the locations of these m cells are independent and identically distributed over N cells. On average, $\frac{ms}{N}$ out of s compromised nodes are detection cells and they will provide the encryption keys. Hence, the FEPL of this scheme is simply

$$p_f^1(m, s) = 1 - (ms/N)/m = 1 - s/N$$

Note that this formula holds after the attacker has compromised s cells and cannot compromise any more cells. We do not consider the FEPL during the process of compromising s cells.

Because all information about one event is stored in one location, Scheme I is subject to single point of failure. Furthermore, both the traffic load and resources for storing the information are not uniformly distributed among all the nodes.

B.2 Scheme II: Time-based Mapping

In this scheme, all nodes store event E occurring in the same time interval T (including a start time and an end time, the duration is denoted as $|T|$) into the same location (L_r, L_c) based

on a group-wide shared key K_T .

$$L_r = H(0|K_T|E|T) \text{ Mod}(N_r). \quad (2)$$

Similarly, $L_c = H(1|K_T|E|T) \text{ Mod}(N_c)$. In addition, every sensor node maintains a timer which fires periodically with time period $|T|$. When its timer fires, a node derives the next group key $K_T = H(K_T)$. Finally, it erases the previous key K_T .

Type II Query: A MS can answer the the following query with one message: *what is about the event E during the time interval T ?* This is because the information about E in T is stored in one location. A MS first determines the location based on K_T, E, T , and then sends a query to it to fetch the data.

Security and Performance Analysis: Due to the use of the one-way hash function, an attacker cannot derive old group keys from the current group key of a captured node. Hence, the locations for storing the events occurred during the previous time periods are not derivable. An attacker has to randomly guess the previous storage cells and detection cells for the event of his interest. The BEPL $p_b^2(m, s)$ of the previous data is very complicated to derive because it depends on the spatial and temporal distribution of m detection cells, the number of previous storage cells for the event, which in turn depends on the number of previous key updating periods and the probability of hash collisions. For ease of analysis, we ignore the case where a cell serves as both a detection cell and a storage cell. Under this assumption, on average an attacker can correctly guess s/N fraction of detection cells and s/N fraction of storage cells. Only when these detection cells are mapped to these storage cells can the attacker decrypt the encrypted data. As such,

$$p_b^2(m, s) = 1 - (s/N)(s/N) = 1 - \left(\frac{s}{N}\right)^2$$

Consider the case $s = 40$ and $N = 400$, the BEPL of Scheme II is 99%. From Fig. 2 we can see the BEPL of scheme I under the same condition is slightly over 90%. Thus, Scheme II provides higher BEPL (i.e., higher backward privacy) than Scheme I.

There are two cases for the FEPL. If the attacker changes the code of the compromised nodes such that in the future these nodes keep their detected event data locally, the FEPL $p_f^2(m, s)$ of this scheme is simply $1 - s/N$. However, if the compromised nodes follow our protocol and hence do not keep a local copy of their data, the FEPL will increase. This is because in the future the event data might be forwarded to new storage cells that are not controlled by the attacker (who is assumed not to be able to compromise more than s cells). Consider that every storage cell used in the future might have been compromised with probability s/N , in this case the FEPL $p_f^2(m, s)$ is the same as the BEPL, i.e., $p_f^2(m, s) = p_b^2(m, s) = 1 - \left(\frac{s}{N}\right)^2$.

Compared to Scheme I, both the traffic load and resources for storing the information in Scheme II are more uniformly distributed in all the cells.

B.3 Scheme III: Cell-based Mapping

In this scheme, all nodes in the same cell $L(i, j)$ store the same type of event E occurring during a time interval T in the same location (L_r, L_c) , based on a cell key K_{ij} shared among

all the nodes in the cell $L(i, j)$. Here

$$L_r = H(0|i|j|E|K_{ij}|T) \text{ Mod}(N_r), \quad (3)$$

and L_c is computed similarly. This scheme differs from the previous schemes in two aspects. First, in this scheme every node in cell $L(i, j)$ updates the cell key K_{ij} periodically based on H such as $K_{ij} = H(K_{ij})$, and then erases the old cell key to achieve backward event privacy. Second, since cell keys are also used for encryption, the updating of cell keys leads to the change of encryption key for the same event detected by the same cell but in different time periods.

Type III Query: A MS can answer the following query with one message: *has event E happened in cell $L(i, j)$ during the time interval T ?* A MS first determines the location based on the key K_{ij} , T , E , and the detection cell $L(i, j)$ of interest, then sends a query to the cell to fetch the data.

Security and Performance Analysis: The updating of cell keys prevents an attacker from deriving old cell keys based on the current cell key of a compromised cell. Hence, the event data recorded in the previous periods are indecipherable irrespective of the number of compromised cells (however, the network controller still keeps the older keys to decrypt previous event data). In other words, the BEFL of this scheme is

$$p_b^3(m, s) = 1$$

Clearly, Scheme III provides the highest BEFL.

The FEPL $p_f^3(m, s)$ of this scheme is the same as in the Scheme II. It can also be seen that this scheme is the least subject to the single point of failure problem compared to the previous schemes. Moreover, both the traffic load and resources for storing the information are the most uniformly distributed among all the nodes.

Summary of the Mapping Schemes Above we have presented three sensor data-to-location mapping schemes with increasing privacy and complexity. These three mapping schemes certainly do not exhaust the design space, because we have three dimensions (time, space, and key) to manipulate. We can easily introduce a row-based (column-based) mapping scheme, where all nodes in the same row i (or column) store the same type of event E occurring during T in the same location (L_r, L_c) . In general, the higher the event privacy, the larger the message overhead for query. However, theoretically the average communication overhead for the detection cells to forward sensor data to the storage cells should be the same in all these schemes as well as in the non-secure DCS systems, owing to the randomness of the storage locations determined by the hash function H . On the other hand, these schemes may be used simultaneously based on the levels of privacy required by different types of data.

C. Improving the Query Efficiency

We have shown that the proposed mapping schemes are capable of answering queries of different granularity and can achieve different levels of privacy. Better privacy is normally achieved at the cost of larger query message overhead. For example, to answer a query like “*Where were the elephants in the last three days*”, one query message is enough in the group-key-based

mapping; however, this may take multiple query messages in the cell-based mapping as the data are stored at multiple places. Next we propose techniques to reduce the query message overhead.

C.1 The Basic Scheme

Suppose a mobile sink (MS) needs to send multiple query messages to multiple storage cells to serve a query. Due to the randomness of the mapping function, these storage cells may be separated by other cells. In the basic scheme, as shown in Figure 3(a), the MS sends one query message to each cell using a routing protocol such as GPSR [5]. Since each query message contains the query information and the *id* of the destination storage cell, these query messages are different and have to be sent out separately. It is easy to see that this scheme has very high message overhead.

Another weakness of the basic scheme is its lack of *query privacy*. Query privacy is measured by the probability that an attacker *cannot* find the *ids* of the storage cells from eavesdropped MS query messages. In the basic scheme, since the MS has to specify the *ids* of the destination storage cells, the query privacy of this scheme, denoted by P_1 , is $P_1 = 0$.

C.2 The Euclidean Steiner Tree (EST) Scheme

A natural solution to reduce the message overhead of the basic scheme is to organize the storage cells as a minimum spanning tree. In this way, the MS can first generate the minimum spanning tree which includes all the storage cells, and then send the query message to these cells following this minimum spanning tree. Although this solution increases the message size, it greatly reduces the number of query messages. Because a message includes many redundant header information, combining multiple messages can significantly reduce the overall message overhead. Similar to the basic scheme, the MS has to include the *ids* of the destination storage cells in his query messages. Thus, the query privacy of this solution is still 0.

To further reduce the message overhead, we can use Euclidean Steiner Tree (EST) [18], which has been shown to have better performance than minimum spanning tree and is widely used in network multicasting. Figure 3(b) shows an EST, which includes some cells other than the storage cells, called *Steiner cells*. Note that these Steiner cells can also help improve the query privacy because they add noise into the set of storage cells.

With EST, the cell that the MS resides will be the root cell. The MS constructs a query message, which contains the *ids* of the cells in the EST, and sends it to its children cells using routing protocols such as GPSR. When a cell head receives a query message, it reconstructs an EST subtree by removing some information such as its own *id* and the *ids* of its sibling nodes, and only keeping the information about the subtree rooted at itself. Then it forwards the query message with the EST subtree to its child cell. This recursive process continues until each storage cell in the EST receives the query message.

To construct an EST, we use a technique proposed by Winter and Zachariasen [18]. Since their solution may return a non-integer Steiner cell, we use the nearest integer Steiner cell to

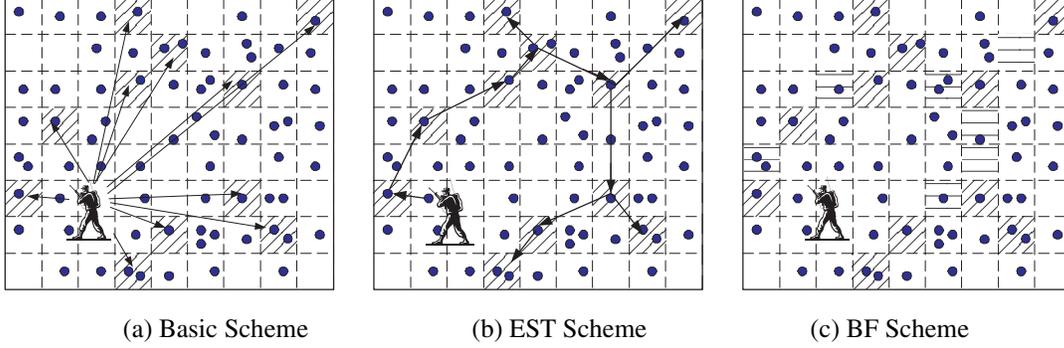


Fig. 3. Three schemes for delivering a query to the storage cells

replace the non-integer steiner cell. Let n denote the number of storage cells. With this solution, an EST spanning k ($2 \leq k \leq n$) cells, has at most $k - 2$ integer Steiner cells, which means that at most $2k - 2$ cells are included in the Steiner tree. The use of Steiner cells can improve the query privacy to at most $1 - \frac{n}{2n-2} = \frac{n-2}{2n-2}$. That is,

$$P_1 = 0 \leq P_2 \leq \frac{n-2}{2n-2} \quad (4)$$

C.3 The Keyed Bloom Filter Scheme

Bloom Filter: A Bloom Filter [19] is a popular data structure used for membership queries. It represents a set $S = s_1, s_2, \dots, s_n$ using k independent hash functions h_1, h_2, \dots, h_k and a string of m bits, each of which is initially set to 0. For each $s \in S$, we hash it with all the k hash functions and obtain their values $h_i(s)$ ($1 \leq i \leq k$). The bits corresponding to these values are then set to 1 in the string. To determine whether an item s' is in S , bits $h_i(s')$ are checked. If all these bits are 1s, s' is considered to be in S .

Since multiple hash values may map to the same bit, Bloom Filter may yield false positives. That is, an element is not in S but its bits $h_i(s)$ are collectively marked by elements in S . If the hash is uniformly random over m values, the probability that a bit is 0 after all the n elements are hashed and their bits marked is $(1 - \frac{1}{m})^{kn} \approx e^{-\frac{kn}{m}}$. Therefore, the probability for a false positive is $(1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-\frac{kn}{m}})^k$. The right hand side is minimized when

$$k = \ln 2 \times m/n, \quad (5)$$

in which case it becomes $(\frac{1}{2})^k = (0.6185)^{m/n}$.

A Bloom Filter can be used to construct query messages. A basic approach is as follows: After an MS determines the location information of all the storage cells, it builds a Euclidean Steiner tree (EST) and gathers the ids of *all* the cells covered by the tree. The MS then inserts the ids into a Bloom Filter, which is sent with other query information to the root cell of the EST using the GPSR algorithm (as shown in Figure 3 (c)). When a query message arrives at a cell, the cell checks the embedded Bloom Filter to determine if its neighbors are in the Bloom Filter, and then forwards the message to them. Recursively, every storage cell receives one query message.

Using Bloom Filter for directed forwarding provides higher query privacy than EST. This is because Bloom Filter introduces some additional noise cells, including the non-storage cells connecting the steiner cells in the EST and a small number of noise cells caused by the false positive rate.

Keyed Bloom Filter: In the Bloom Filter-based scheme, an attacker can freely check if a cell is one of the storage cells although there could be a high false positive rate. To further improve the query privacy, we should disable the attacker's capability in performing membership verification over a Bloom Filter. This motivates our design of a keyed Bloom Filter (KBF) scheme, which uses cell keys to "encrypt" the cell ids before they are inserted. In this way, an attacker can derive none or only a small number of cell ids from a query message. This ensures that the attacker has negligible probability to identify the storage cells other than random guessing.

In the KBF scheme, each cell id is concatenated with the cell key of its parent node in the EST before it is inserted into the Bloom Filter. Specifically, to insert cell id x , the bits corresponding to $H_i(x|k_p)$ ($i = 1, \dots, k$) are set to 1, where k_p is the cell key of the parent of cell x . When a query message arrives at a cell, the cell concatenates its own cell key with the id of each neighboring cell that is not a neighbor of its own parent node (to avoid redundant computations and forwarding), and determines whether the neighbor is in the Bloom Filter. If it is, the message is forwarded to the neighbor. Algorithm 1 and Algorithm 2 formally describe the ways to create a Bloom Filter and to forward a query message, respectively.

Algorithm 1 Create a Bloom Filter

Input: an array of storage-cell Cartesian coordinates $c[]$;

Output: Bloom Filter BF ;

Procedure:

- 1: initialize a Bloom Filter BF ;
 - 2: build Steiner tree based on $c[]$;
 - 3: **for** each cell u in the Steiner tree **do**
 - 4: $p =$ parent of u ; $k_p =$ cell key of p ;
 - 5: map $(u|k_p)$ into BF ;
 - 6: **end for**
 - 7: return BF ;
-

Query Privacy: In this scheme, cell ids are "encrypted" with cell keys before being inserted into the Bloom Filter. If an attacker has not compromised any cells in the EST, he will not know any cell keys. In this case, he cannot obtain any information about storage cells from an eavesdropped query message. Next we consider the case that the attacker has compromised

Algorithm 2 Forward a Query Message

Input: a query message received by cell u , which includes a Bloom Filter BF .
Procedure:
1: $k_u =$ cell key of u ;
2: **for** each neighboring cell u' of u **do**
3: **if** $u' \neq$ parent of $u \wedge u' \neq$ neighbor of the parent of $u \wedge BF$ contains u' **then**
4: forward the query message to u'
5: **end if**
6: **end for**

some cells in the EST. If a compromised cell is contained in the EST, from the received query message it can find out its neighboring cells that belong to the EST. However, it cannot verify the membership of the other cells. In fact, this is one prominent advantage of the KBF scheme over the EST scheme. To make the EST scheme more secure, a straightforward extension would be to encrypt the EST tree. To enable every cell in the tree to access the information for correct forwarding of a query message, a group key will need to be used to encrypt the EST tree. Thus, an attacker can decrypt the entire EST as long as he can compromise one cell. Clearly, the KBF scheme offers much better query privacy than the EST scheme. The query privacy of the KBF scheme and other schemes are compared in Section IV, and the results show that the KBF scheme has the highest privacy.

C.4 Plane Partition

The EST scheme reduces the number of query messages at the price of larger message size. The limited packet size, e.g., 29 bytes in TinyOS [20] may prevent the MS to piggyback all the storage cell ids together with the query information in a single packet. A Bloom Filter may be designed to fit in a packet, but to maintain a low false positive rate, only a limited number of cell ids should be included in a packet. To address this problem, we use multiple Steiner trees, each of which is encoded into a single packet. Because partitioning a Steiner tree into multiple Steiner trees, known as the minimum forest partition problem, is NP-hard ([21]), we propose heuristics to perform the partition.

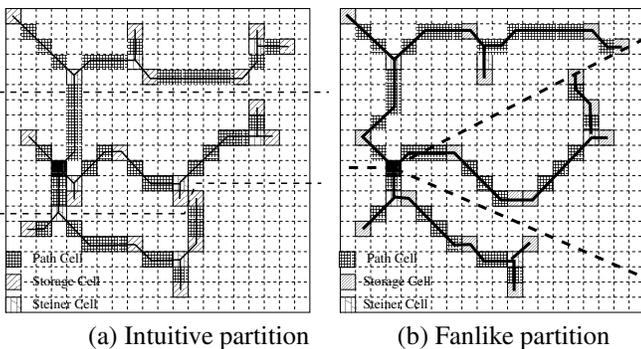


Fig. 4. 17 storage cells are partitioned into three parts

In Figure 4 (a), the solid lines are used to represent the EST tree, and the shaded areas along these solid lines are used by Bloom Filters to encode the EST tree. An intuitive partition method is to first cluster the storage cells in a top-down and left-right fashion, and then build a sub-EST within each partition.

We can let the EST scheme and the KBF scheme have the same partitions and build the same sub-EST trees. After the partition, the MS sends a query to each partition at the same time. In this way, the message size can be reduced. Further, since multiple queries are sent out at the same time, the average query delay is also reduced.

Fanlike Partition Method: With the intuitive partition, the query message from the MS has to go through some redundant cells. For example, in Figure 4 (a), the query message of the MS has to go through many cells before reaching the top partition. To address this problem, we change the Cartesian coordinates into Polar coordinates. In this new coordination system, storage cells are within $[-\pi, \pi]$. The partition algorithm scans the plane from $-\pi$ to π and collects enough storage cells into each partition. Figure 4 (b) shows one example of dividing the plane into three partitions using the Fanlike partition method.

IV. PERFORMANCE EVALUATIONS

In this section, we evaluate and compare the performance of three query schemes: the Basic scheme, the Euclidean Steiner Tree (EST) scheme and the Keyed Bloom Filter (KBF) scheme. In our simulation setup, each query message contains the query information and the encoded query path. The query information occupies 4 bytes which are used to represent time and event², and 25 bytes are used to represent the query path. For evaluation purpose, we do not consider the overhead of source authentication.

In the EST scheme, the query path is encoded as a Steiner tree. Each node id is presented by two bytes, so only 12 cell ids can be encoded in each packet. In the KBF scheme, 25 bytes are used to encode the query path with Bloom Filter, and it is expected to achieve an acceptable false positive rate, say 0.1. Considering these limitations, we choose $(n, k) = (20, 5)$.

These schemes are evaluated under various storage cell densities, ranging from $\frac{1}{40}$ to $\frac{1}{2.5}$. The *storage cell density* is defined as the ratio of the number of storage cells to the number of total cells in the plane. For example, with our setting of 20×20 cells, a density of $\frac{1}{10}$ means that there are about $400 * \frac{1}{10} = 40$ storage cells.

Four metrics are used to evaluate the performance of the proposed schemes: the number of query messages, the average query delay, the maximum query delay and the message overhead. The number of query messages is the total number of messages sent out by the MS for a query. The average query delay is the average of the query delays for different storage cells. The maximum query delay is the maximum among all the query delays. The message overhead is defined as the total number of transmitted hops of all the messages sent out by the MS to serve a query. In the KBF scheme, the message overhead also includes the extra messages due to false positive.

A. Choosing the Partition Method

In this subsection, we evaluate the performance of EST with intuitive partition and EST with Fanlike partition. As shown in

²Some applications may require more bytes; nevertheless, since we are interested in the comparative results of multiple schemes, normally the payload size will not affect much. Further, the time should be in hour/minute level instead of microsecond level, and hence only need less number of bits.

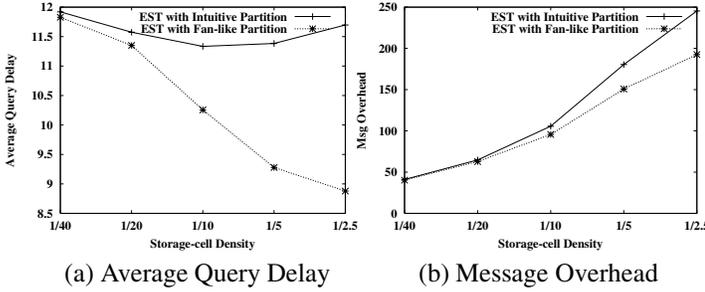


Fig. 5. comparisons between different partitions

Figure 5, the Fanlike partition method outperforms the intuitive method in terms of query delay and message overhead. We do not show the number of messages, since both schemes have the same number of messages determined by the packet size.

As discussed earlier, in the intuitive partition method, each query message is sent from the MS to the partition, which may go through many redundant cells and hence increase the message overhead. However, in the Fanlike partition, less redundant cells are involved, and hence the message overhead is lower. This also explains why the Fanlike partition has lower average and maximum query delay when compared to the intuitive partition.

In Figure 5 (a), with Fanlike partition, the average query delay drops as the storage cell density increases. This can be explained as follows. When the storage cell density is high, each partition is small. Therefore, the Steiner tree is limited within a small range and the zig-zag paths from MS to storage cells tend to be shorter. This results in smaller average query delays.

We also evaluated the performance of the KBF scheme under both partition methods. The results are similar to EST where the Fanlike partition performs better. Thus, we use the Fanlike partition method in the following comparisons.

B. Performance Comparisons of Different Schemes

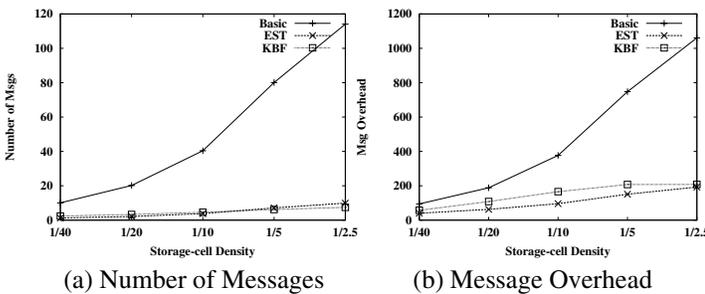


Fig. 6. The message overhead of different schemes

Figure 6 compares the number of messages and the message overhead of the three schemes: the Basic scheme, the EST scheme and the KBF scheme. As can be seen, both optimization schemes (EST and KBF) outperform the basic scheme since the optimization schemes combine several messages into one. We can also see that the message overhead of the KBF scheme is higher than the EST scheme although both schemes have similar number of messages. This is due to the fact that the query messages in the KBF scheme may go through some redundant cells due to false positive.

Figure 7 (a) (b) compares the average delay and the maximum delay of the three schemes. As can be seen, the basic scheme outperforms the other two. This is because in the basic scheme, the query messages are sent directly to the storage cells in parallel along shortest paths, resulting in a lower query delay. Although EST and KBF can reduce the message overhead, the query delay is increased since the message has to go through many intermediate cells sequentially.

As shown in Figure 7(a) and (b), when the storage cell density is low, KBF outperforms EST in terms of query delay. To explain this, we need to understand the effects of the number of partitions. When the number of partitions is small and hence each partition is large, the path to each storage cell is more zig-zag like, which may result in long delay. As shown in Figure 6 (a), when the density is low, EST has less number of messages and hence less number of partitions, which means that EST will have large partitions and long delay. Similarly, when the density is high, EST has more partitions and shorter delay.

In addition, as shown in Figure 7(c), the KBF scheme has the highest query privacy. Even after $s = 20$ cells have been compromised, the query privacy level is still above 83%.

In summary, there is a tradeoff among query delay, message overhead, and query privacy. The Basic scheme has the lowest delay but the highest message overhead and the lowest query privacy. The EST scheme and the KBF scheme can significantly reduce the number of messages and the message overhead with the same level of query delay. Moreover, the query privacy level of KBF is far higher than the other schemes.

V. RELATED WORK

There are mainly two approaches for restricting mobile sink (MS) access to sensor data: policy enforcement and data perturbation. In the spirit of the first approach, Myles et al. [22], Hengartner and Steenkiste [23] studied the issue of specifying location privacy policies on which access control decisions are based. Alternatively, anonymity mechanisms could also be employed to provide the required level of privacy by properly perturbing the sensor data before its release. Gruteser et al. [24] proposed techniques such as data cloaking and hierarchical data aggregation to prevent an attacker from tracking the precise location of an individual monitored by sensors. The main difference between our work and the previous work is that we achieve sensor data privacy in an unattended environment by encryption and random location mapping, not by policy enforcement or data perturbation. These techniques are complementary to each other and can be applied jointly if needed.

Location-based forwarding has been studied for both mobile ad hoc networks and sensor networks. The location-aided routing [25] was proposed to reduce the cost of discovery by restricted area flooding when the uncertainty about a destination is limited. Greedy routing schemes, e.g., GPSR [5], choose the next hop that provides most progress towards the destination. In these schemes, the delivery of packets is guaranteed by planarizing the network graph and applying detour algorithms which avoid obstacles using the “right hand rule” strategy. Niculescu and Nath [26] proposed trajectory-based routing, in which the source encodes trajectory to traverse and embeds it into each

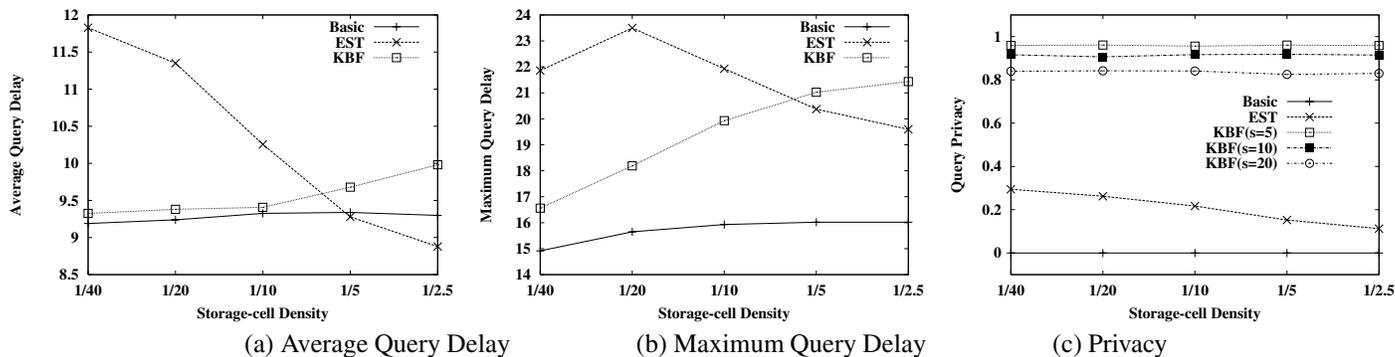


Fig. 7. Comparisons among different schemes

packet. Upon the arrival of each packet, intermediate nodes employ greedy forwarding techniques such that the packet follows its trajectory as much as possible. With this scheme, routing becomes source-based while there is no need for maintaining routing tables at intermediate nodes. We note that the scheme in [26] is suitable for a regular shape trajectory, not for totally random shape trajectory, which is the case in p DCS.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed solutions on privacy support for data centric sensor networks (p DCS). The proposed schemes offer different levels of location privacy and allow a tradeoff between privacy and query efficiency. p DCS also includes several query optimization techniques based on Euclidean Steiner Tree and Bloom Filter to minimize the query message overhead and increase the query privacy. Simulation results verified that the KBF scheme can significantly reduce the message overhead with the same level of query delay. More importantly, the KBF scheme can achieve these benefits without losing any query privacy.

To the best of our knowledge, this is the first paper to address privacy issues in data-centric sensor networks. As the initial work, we do not expect to solve all the problems. In the future, we will address other issues such as source anonymity, key management, and look into other query techniques to balance the tradeoff between query delay and message overhead.

ACKNOWLEDGMENT

This work was supported in part by Army Research Office (W911NF-05-1-0270) and the National Science Foundation (CNS-0524156, CNS-0627382).

REFERENCES

- [1] S. Ratnasamy, B. karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," *ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.
- [2] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensor networks," *ACM SIGCOMM Computer Communication Review archive*, vol. 33, no. 1, pp. 137–142, 2003.
- [3] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin and J. Heidemann, "Multi-resolution Storage and Search in Sensor Networks," *ACM Transactions on Storage*, August 2005.
- [4] W. Zhang, G. Cao, and T. La Porta, "Data Dissemination with Ring-Based Index for Wireless Sensor Networks," *IEEE International Conference on Network Protocols (ICNP)*, pp. 305–314, November 2003.
- [5] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *ACM Mobicom*, 2000.
- [6] A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," *Communications of the ACM*, vol. 47, no. 6, June 2004.
- [7] J. Deng, C. Hartung, R. Han, and S. Mishra, "A practical study of transitory master key establishment for wireless sensor networks," in *IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, 2005, pp. 289–299.
- [8] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "Spins: security protocols for sensor networks," in *ACM Mobicom*, 2001.
- [9] W. Du, J. Deng, Y. Han, and P. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003, pp. 42–51.
- [10] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *ACM CCS*, 2003.
- [11] S. Zhu, S. Setia, and S. Jajodia, "Leap: Efficient security mechanisms for large-scale distributed sensor networks," in *ACM CCS*, 2003.
- [12] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, Feb. 2006.
- [13] Hao Yang, Fan Ye, Yuan Yuan, Songwu Lu, and William Arbaugh, "Toward resilient security in wireless sensor networks," in *ACM MOBIHOC*, 2005.
- [14] W. Zhang and G. Cao, "Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration-Based Approach," *IEEE INFOCOM*, March 2005.
- [15] J. Deng, R. Han, and S. Mishra, "Intrusion tolerance and anti-traffic analysis strategies for wireless sensor networks," *International Conference on Dependable Systems and Networks (DSN'04)*, June 2004.
- [16] C. Ozturk, Y. Zhang, and W. Trappe, "Source-location privacy in energy-constrained sensor networks routing," *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04)*, October 2004.
- [17] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.
- [18] Pawel Winter and Martin Zachariasen, "Euclidean steiner minimum trees: An improved exact algorithm," *Networks*, vol. 30, no. 3, pp. 149–166, 1997.
- [19] B. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communications of the ACM*, 1970.
- [20] "The tinydb project," <http://telegraph.cs.berkeley.edu/tinydb/>.
- [21] Roberto Cordone and Francesco Maffioli, "On the complexity of graph tree partition problems," *Discrete Appl. Math.*, vol. 134, no. 1-3, pp. 51–65, 2004.
- [22] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," in *IEEE Pervasive Computing*, 2003.
- [23] U. Hengartner and P. Steenkiste, "Protecting access to people location information," in *Proceedings of the First International Conference on Security in Pervasive Computing*, 2003.
- [24] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald, "Privacy-aware location sensor networks," in *Proceedings of 9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX)*, 2003.
- [25] Y. Ko and N. Vaidya, "Location-aided Routing in Mobile Ad Hoc Networks," *ACM Mobicom*, pp. 66–75, 1998.
- [26] D. Niculescu and B. Nath, "Trajectory Based Forwarding and Its Applications," *ACM MOBICOM*, 2003.