

# Securing Distributed Data Storage and Retrieval in Sensor Networks\*

Nalin Subramanian, Chanjun Yang, and Wensheng Zhang  
Department of Computer Science  
Iowa State University, Ames, IA 50011  
Email: {nvsbram, cjyang, wzhang}@cs.iastate.edu

## Abstract

*Sensor networks have been an attractive platform for pervasive computing and communication. Due to the lack of physical protection, however, sensor networks are vulnerable to attacks if deployed in hostile environments. When a sensor network is under attack, the most fundamental concern is that information communicated or stored in the network remains safe. The past research has focused on securing information in communication, but how to secure information in storage has been generally overlooked. Meanwhile, distributed data storage and retrieval have become popular for efficient data management in sensor networks, which renders the absence of schemes for securing stored information to be a more severe problem. Hence, we propose in this paper three schemes to deal with the problem. All the schemes have the following properties: (i) Only authorized entities can access data stored in the sensor network; (ii) The schemes are resilient to a large number of sensor node compromises. The second and the third schemes do not involve any centralized entity except for a few initialization or renewal operations, and thus support secure, distributed data storage and retrieval. The third scheme further provides high scalability and flexibility, and hence is most suitable in real applications. The effectiveness and efficiency of the proposed schemes have also been verified through extensive analysis and TOSSIM-based simulations.*

## 1. Introduction

The sensor network [1, 2] is a collection of small-size, low-power, low-cost sensor nodes that have some computation, communication, storage and even movement capabilities. These nodes can operate unattended, sensing the environment, generating data, processing data, and providing the data to users. With these features, sensor networks have

been adopted in many pervasive computation and communication scenarios such as remote surveillance, habitat monitoring, and so on [3, 4]. Along with the attractive features and the increasingly important roles in applications, the sensor network however has its own limitations such as lack of physical protection, which is caused by its unattended deployment environment and absence of tamper resistance. Without physical protection, the sensor network may suffer from *node compromise-based attacks*, besides the attacks (e.g., eavesdropping, jamming, etc.) that can happen in any type of wireless networks. For instance, attackers invading the deployment field of a sensor network can capture and compromise sensor nodes, and then steal the information stored in the nodes.

The past research on sensor network security has focused on securing the information in communication; in particular, a large number of key establishment [5, 6, 7], message authentication [8, 9] and intrusion detection [10] schemes have been proposed for the purpose. However, securing information in storage has not received adequate attention from the research community. Meanwhile, more and more distributed in-network data storage and retrieval schemes [3, 11, 12, 13, 14, 15, 16, 17, 18] have been proposed for efficient data management, which makes the absence of mechanisms for securing stored information to be a more severe issue. In the distributed in-network data storage schemes such as TTDD [3], DCS [11, 13], TSRA [17], the landmark-based information brokerage system [18], and others [12, 14, 15, 16], after a sensor node has generated some data, the node stores the data locally or at some designated nodes in the network, instead of immediately forwarding the data to a centralized data center, which is usually located out of the network; the locally stored data are sent out from the in-network storages only when they are queried. To protect the stored data from being stolen by the attackers who have compromised storage nodes, the data should be encrypted. At the same time, users authorized to access the data should be able to decrypt the data. This raises the issue of how to *simultaneously* achieve *data security* and *data accessibility* in distributed data storage and

\*This work was partially supported by NSF CYBERTRUST-0627354.

retrieval systems.

If every data query goes through a trustworthy centralized entity, the above issue can be addressed based on the following idea: the locally stored data are encrypted with some encryption keys that will not be exposed to unauthorized users; the centralized entity keeps track of the encryption keys, and perform data decryption to facilitate authorized users in accessing the data. However, the frequent involvement of a centralized entity may compromise the efficiency and effectiveness of the system. For example, let us assume a sensor network is deployed to monitor a battlefield [3]. The authorized users of the sensor data (e.g., soldiers) may move in the field and issue queries from any location in the field. If every query must go through a centralized entity, data access delay could be significantly increased. Besides, the query or data response could be lost due to link failures, traffic congestion, or other reasons. The loss of data and the data access delay may impair the vision of the soldiers, which could be life-threatening in some critical scenarios. Even worse, maintaining a centralized entity in the hostile environment could be infeasible since the centralized entity would become an attractive target for attacks.

To minimize the involvement of any centralized entity in data retrieval, decentralizing data retrieval is desired, and this has in fact been realized in a number of distributed in-network data storage systems [3, 11, 12, 18] which provide some information brokerage mechanisms to bridge data producers and consumers. The introduction of distributed retrieval, however, has made the problem of securing distributed data storage to be more challenging. To support both distributed data storage and distributed data retrieval, the following requirements should be met simultaneously: (i) only authorized entities can access data stored in the network; (ii) the storage and retrieval operations should involve any centralized entity at least as possible; (iii) the scheme should be resilient to the compromises of a large number of sensor nodes.

In this paper, we propose a series of schemes for securing both distributed data storage and distributed data retrieval. We first propose a *Simple Hash-Based (SHB) scheme*, which supports distributed data storage but relies on a centralized entity for data retrieval. To further enable and secure distributed data retrieval, we propose an *Enhanced Hash-Based (EHB) scheme*, in which secret keying information is preloaded to sensor nodes and authorized users, and then they can perform distributed data storage and retrieval without the involvement of any centralized entity. However, the EHB scheme imposes strict restrictions on the network and is not scalable since the network life time should be determined before node deployment and be fixed after that; the overhead for storing preloaded information is linear to the system life time. To remove the restrictions and improve scalability, we finally propose an *Adaptive Polynomial-Based (APB) scheme*, which allows a network controller to refresh the keying information stored in sensor nodes on demand, and thus provide better scala-

bility and flexibility; meanwhile, the network controller is only involved occasionally, and thus does not noticeably affect the system efficiency. The security properties and the efficiency of the proposed schemes are evaluated through extensive analysis and TOSSIM-based simulations.

The rest of the paper is organized as follows: Section II presents the network assumptions and attack model. Section III presents the proposed schemes. This is followed by the security analysis and performance evaluation in Section IV and V, respectively. Section VI finally concludes the paper.

## 2 Preliminaries

### 2.1. Network assumptions

We consider a sensor network that is composed of a network controller, which could be online or offline, and a large number of sensor nodes, each has a unique ID. Following the previous works (e.g., TTDD [3], DCS [11], GEM [13], TSRA [17], etc.) on in-network distributed storage, we assume each sensor node has some capacity to store sensor data. Once these sensor nodes have generated some data, the data are stored locally or at some designated storage sensor nodes. The stored data can be retrieved by authorized users, e.g., soldiers in a battlefield, tourists in a national park, or zoologists in the wilder. To simplify the presentation of our schemes, we assume that sensor data are retrieved based on time (The schemes can be extended to support more complicated modes of data retrieval).

We also assume the sensor nodes maintain loose time synchronization, which is in fact a prerequisite for many applications. Based on time synchronization, the lifetime of the network is divided into phases, each having the same time duration denoted as  $\tau$ . Therefore, Phase 0 spans from time 0 to time  $\tau$ , Phase 1 spans from time  $\tau$  to time  $2\tau$ , and so forth.

### 2.2. Attack model and Design Goal

The distributed in-network data storage system could be attacked in many ways. In this paper, we focus on the attack that unauthorized users attempts to steal data stored in the network. Concretely, the attack can be conducted as follows: the attacker first compromises some sensor nodes; it then captures the data stored in the sensor nodes; if the data are encrypted, it will attempt to figure out the key and thus decrypt the data.

Due to the lack of physical protection for sensor nodes, the compromise of sensor nodes and the capturing of data from the compromised sensor nodes cannot be fully prevented. The moderate objective of our design is to prevent the attacker from interpreting the data they have captured, through appropriate strategy of data encryption; meanwhile, our design aims to not noticeably affect the efficiency for authorized users to retrieve and interpret the data.

### 3. Proposed Schemes

In this section, we propose a series of schemes for securing distributed data storage and retrieval.

#### 3.1. Simple Hash-Based (SHB) Scheme

The SHB scheme has three components, initialization, data storage (encryption), and data retrieval (decryption), which are detailed as follows:

##### (1) System Initialization

Initially, the network controller picks a secure hash function, denoted as  $h(\cdot)$  and a master key denoted as  $K_m$ . Before deploying a sensor node whose ID is  $u$  (We call the node  $u$  thereafter), the network controller preloads to the node hash functions  $h(\cdot)$ , and an initial data encryption key  $K_{u,0}$ , which is computed as  $h(K_m | u)$ . Here,  $|$  stands for the concatenation operator. After being deployed, sensor node  $u$  establishes a sensor data item counter  $c$ , and initializes it to 0.

##### (2) Data Storage

When node  $u$  generates a data item (denoted as  $D$ ), it performs the following steps:

- (i) The data item is encrypted with the current encryption key  $K_{u,c}$ . Then, the encrypted version is stored with the current value of  $c$  and the current time stamp  $t$ . That is, a 3-tuple  $\langle \{D\}_{K_{u,c}}, c, t \rangle$  is stored, where  $\{D\}_{K_{u,c}}$  represents the result of encrypting data item  $D$  with key  $K_{u,c}$ .
- (ii) The counter  $c$  and the data encryption key are updated. That is,  $c \leftarrow c + 1$ , and the current data encryption key becomes  $K_{u,c} = h(K_{u,c-1})$ .
- (iii) The previous data encryption key  $K_{u,c-1}$  is erased.

##### (3) Data Retrieval

When a user authorized to query sensor data generated during certain time period, the user sends out query. For any response data sent by any sensor node  $u$ , denoted as  $\langle \{D\}_{K_{u,c}}, c, t \rangle$ , the user has to resort to the network controller, which keeps master key  $K_m$  and can compute  $K_{u,c}$  as  $h^c(K_m | u)$ , to decrypt  $\{D\}_{K_{u,c}}$  and thus obtain the data item  $D$ .

**Discussion** The SHB scheme requires the network controller to be tightly involved in every data query for data decryption. The interaction may incur high communication overhead especially when the user is far away from the controller. The controller could become a communication bottleneck if there exist many users. These limitations motivate us to design new schemes that can secure both distributed data storage and distributed data retrieval.

#### 3.2. Enhanced Hash-Based (EHB) Scheme

Assume the network lifetime has  $n$  phases. The EHB scheme is detailed as follows:

##### (1) System Initialization

Similar to SHB, the network controller picks a secure hash function, denoted as  $h(\cdot)$  and a master key denoted as  $K_m$ . Before deploying every sensor node  $u$ , the network controller preloads to the node hash functions  $h(\cdot)$ , and  $n$  initial data encryption keys  $K_{u,i,0}$  ( $i = 0, \dots, n-1$ ) for each phase. Each  $K_{u,i,0}$  is computed as  $h(h(K_m | i) | u)$ .

##### (2) Per-phase Initialization

At the beginning of Phase  $i$  ( $i = 0, \dots, n-1$ ), sensor node  $u$  establishes a sensor data item counter  $c_i$ , and initializes it to 0. Node  $u$  also initializes the current data encryption key to be  $K_{u,i,0}$ .

##### (3) Data Storage

When node  $u$  generates a data item (denoted as  $D$ ), it performs the following steps:

- (i) The data item is encrypted with the current encryption key  $K_{u,i,c_i}$ . Then, the encrypted version is stored with the current phase number  $i$ , the current value of  $c_i$ , and the current time  $t$ . That is, a 4-tuple  $\langle \{D\}_{K_{u,i,c_i}}, i, c_i, t \rangle$  is stored.
- (ii) The counter  $c_i$  and the data encryption key are updated. That is,  $c_i \leftarrow c_i + 1$ , and the current data encryption key becomes  $K_{u,i,c_i} = h(K_{u,i,c_i-1})$ , which is equal to  $h^{c_i}(K_{u,i,0})$ .
- (iii) The previous data encryption key  $K_{u,i,c_i-1}$  is erased.

##### (4) Data Retrieval

When a user authorized to query sensor data generated during certain time period (i.e., from Phase  $i_s$  to Phase  $i_e$ ), the user is preloaded with the following initial keys of phases:  $\{PK_i = h(K_m | i) | i = i_s, \dots, i_e\}$ .

After the user sends out queries, it will obtain responses from sensor nodes. For each response  $\langle \{D\}_{K_{u,i,c_i}}, i, c_i, t \rangle$  ( $i_s \leq i \leq i_e$ ) sent from sensor node  $u$ , the user can recover the data encryption key  $K_{u,i,c_i}$  by computing  $h^{c_i}(h(PK_i) | u) \equiv h^{c_i}(h(K_m | i) | u)$ . Then the data item  $D$  can be decrypted with key  $K_{u,i,c_i}$ .

**Discussion** Comparing to the SHB scheme, the EHB scheme supports both distributed data storage and distributed data retrieval. The network controller is involved only for some initialization operations. However, the EHB scheme has such limitations as the number of phases is fixed and the initial data encryption keys for all these phases

should be preloaded before sensor nodes are deployed. These limitations *impede the scalability* of the network in terms of network life time and storage consumption.

A straightforward way to further extend the EPB scheme is as follows: Each sensor node is preloaded with only the initial data encryption keys for a certain number of phases, and the network controller will disseminate new keys before the existing keys have been used up. This way, the network life time will not be constrained by the number of keys preloaded to each sensor node, and the sensor node does not need to allocate a large storage space for storing the keys in order to support long life time. However, this will require the network controller to disseminate securely new keys to individual sensor nodes. Since each sensor node needs different sets of keys, the overhead for the dissemination is proportional to the number of sensor nodes, the number of keys each node must receive, and the frequency of new key dissemination. Therefore, the solution may not be feasible when the network scale is large, and/or the length of phase is short (and hence the frequency of new key dissemination or the number of keys each node must receive will be large).

### 3.3. Adaptive Polynomial-Based (APB) Scheme

To deal with the limitation of non-scalability in the EHB scheme, we now propose a novel APB scheme. In this scheme, the network controller can disseminate on-demand a single copy of *network-wide seed polynomial* to all sensor nodes; then, each sensor node can refresh their data encryption keys based on the received polynomial. This way, the system life time and the storage overhead at each sensor node can be dynamically adjusted. Note that, this approach is significantly more efficient than disseminating different sets of keys to different sensor nodes (as discussed above on extending EHB).

In the following, we will first present the algorithm for disseminating the network-wide seed polynomial, then the algorithm for individual sensor nodes to derive their data encryption keys and encryption data, and the algorithm for individual users to retrieve and decrypt data they are authorized to access. Finally, an enhancement for further improving the efficiency of seed polynomial dissemination will be described.

#### 3.3.1 Dissemination of Network-wide Seed Polynomial

To enable dissemination of network-wide seed polynomials, the following *system preparation* should be performed before sensor nodes are deployed: The network controller arbitrarily constructs a three variable polynomial  $p(x, y, z)$  called *perturbing polynomial*<sup>1</sup> over a finite field  $F_q$ , where  $q$  is primary number. We further define two integers  $l$  and

<sup>1</sup>This polynomial is used for prevent eavesdropping of the seed polynomial broadcast in the network.

$r$ , where  $l$  is the smallest integer such that  $2^l > q$  (i.e., every element of the finite field  $F_q$  can be represented with  $l$  binary bits), and  $r$  is some integer smaller than  $l$ . Before each sensor node  $u$  is deployed, the node is preloaded with a *perturbed* share of  $p(x, y, z)$ , i.e.,

$$\widetilde{p}_u(y, z) = p(u, y, z) - \alpha_u,$$

where  $\alpha_u$  is an element of  $F_q$  randomly picked from  $\{0, \dots, 2^{r-2} - 1\}$ .

When the Node is deployed, Node  $u$  derives  $m$  ( $m$  is a system parameter) perturbed shares of  $p(u, y, z)$ :

$$\begin{aligned} \overline{p}_u(y, 0) &= \widetilde{p}_u(y, 0) - r_0 \\ \overline{p}_u(y, 1) &= \widetilde{p}_u(y, 1) - r_1 \\ &\dots \\ \overline{p}_u(y, m-1) &= \widetilde{p}_u(y, m-1) - r_{m-1} \end{aligned}$$

where  $r_0, r_1, \dots, r_{m-1}$  are elements of  $F_q$  randomly picked from  $\{0, \dots, 2^{r-2} - 1\}$ . After the node derives  $m$  perturbed shares,  $\widetilde{p}_u(y, z)$  is immediately removed.

Now, suppose the network controller wants to disseminate a network-wide seed polynomial for producing keys for Phases  $i_s, \dots, i_e$ . The following steps will be performed:

- (i) The network controller arbitrarily picks a bivariate polynomial  $f(x, y)$ , called *master polynomial*, which provides the basis for generating data decryption keys for Phases  $i_s, \dots, i_e$ .
- (ii) Based on the master polynomial  $f(x, y)$  and the perturbing polynomial  $p(x, y, v)$  (where the network controller is disseminating *network-wide seed polynomial* for the  $(v+1)^{th}$  time, for  $v \in \{0, \dots, m-1\}$ ), the network controller constructs the network-wide seed polynomial  $w(x, y)$ ; specifically,

$$w(x, y) = f(x, y) + p(x, y, v) + \mu_v,$$

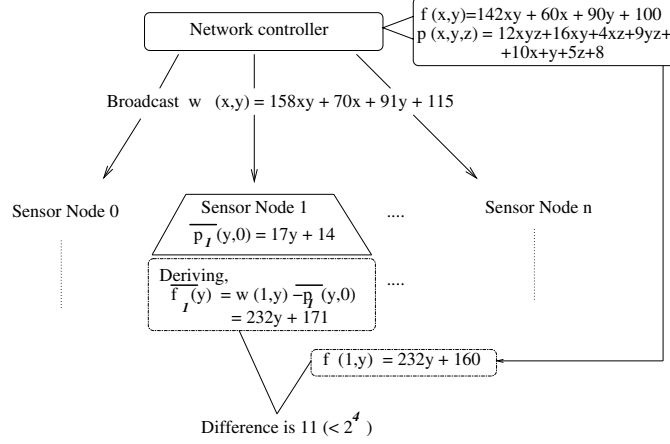
where  $\mu_v$  is an element of  $F_q$  randomly picked from  $\{0, \dots, 2^{r-1} - 1\}$ .

- (iii) The network controller broadcasts polynomial  $w(x, y)$  to all sensor nodes
- (iv) Upon receiving  $w(x, y)$ , every sensor node  $u$  derives *its own seed polynomial*

$$\overline{f}_u(y) = w(u, y) - \overline{p}_u(y, v)$$

Note that,  $\overline{f}_u(y)$  (i.e., the seed polynomial of node  $u$ ) and  $f(x, y)$  (i.e., the network-wide seed polynomial) has the following relationship:

$$\overline{f}_u(y) - f(u, y) \in \{0, \dots, 2^r - 1\}. \quad (1)$$



**Figure 1.** An Example of Disseminating A New Seed Polynomial ( $r = 4, z = 0$ )

This is because

$$\begin{aligned}
 \overline{f}_u(y) &= w(u, y) - \overline{p}_u(y, v) \\
 &= (f(u, y) + p(u, y, v) + \mu_v) \\
 &\quad - (p(u, y, v) - \alpha_u - r_v) \\
 &= f(u, y) + (\mu_v + \alpha_u + r_v),
 \end{aligned}$$

and

$$(\mu_v + \alpha_u + r_v) \in \{0, \dots, 2^r - 1\}.$$

Fig. 1 shows an example to illustrate the above procedure. Let system parameters be  $r = 4$ , new master polynomial is  $f(x, y) = 142xy + 60x + 90y + 100$ , perturbing polynomial is  $p(x, y, z) = 24xyz + 16xy + 4xz + 9yz + 10x + y + 5z + 8$ . Suppose the network controller is disseminating seed for the 1st time, then  $z = 0$ . Suppose the arbitrarily chosen random numbers be  $(\alpha_1 + r_1) = 7$  and  $\mu_0 = 4$  ( $\in \{0, \dots, 2^{4-1} - 1\}$ ) used for  $\overline{p}_1(y, 1)$  and  $w(x, y)$ . Based on disseminated polynomial, node 1 derives  $\overline{f}_1(y) = 232y + 171$ , which differs  $f(1, y) = 232y + 160$  by 11 (within  $\{0, \dots, 2^4 - 1\}$ ).

After the extraction of master seed polynomial, the node  $u$  removes  $\overline{p}_u(y, v)$ . Each time a node receives the network-wide seed polynomial, the node uses the current perturbing polynomial share to extract the master polynomial, followed by deleting the current perturbing share ( $\overline{p}_u(y, v)$ ). And for any of the future network-wide seed polynomial update from network controller, the node uses the next derived perturbing polynomial share ( $\overline{p}_u(y, v + 1)$ ). This way the adversary who can eavesdrop  $w(x, y)$ , will not be able to derive the currently disseminated master seed polynomial by capturing the nodes at later duration. This is because, once the node  $u$  deletes the currently used perturbing polynomial share  $\overline{p}_u(y, v)$  of  $\widetilde{p}_u(y, z)$ , it will be hard for the adversary to calculate the perturbing polynomial share at the time the adversary attacks. Since the node has only  $\{\overline{p}_u(y, v + 1), \dots, \overline{p}_u(y, m - 1)\}$ , it will become harder for the adversary to obtain the original share  $\widetilde{p}_u(y, z)$ .

As to be explained in Section 3.3.2, sensor node  $u$  will derive its initial keys for Phases  $i_s, \dots, i_e$  from  $\overline{f}_u(y)$ . Also, as to be shown in Section 4, with the above procedure for disseminating seed polynomial, any sensor node  $u$  has very low probability to derive the seed polynomial of any other sensor node, if system parameter  $r$  as well as  $t$  the degree of  $x$  and  $y$  in polynomial  $f(x, y)$  are appropriately chosen.

Note that, initially preloaded perturbing polynomial  $\widetilde{p}_u(y, z)$  can be used only for  $m$  times dissemination of network-wide seed polynomial. As a system parameter,  $m$  is set when the perturbing polynomial is preloaded; meanwhile, whenever a network-wide seed polynomial is disseminated, the number of phases for which the seed polynomial is to be used is announced online, which provides the flexibility to accommodate adjustable network lifetime. More flexibility can be achieved if new perturbing polynomials can be disseminated after the old one is used up, and this will be studied in our future work.

Also note that, we do not describe the distribution of keys for the beginning phases (i.e., Phase  $0, 1, \dots$ ). These keys can be directly preloaded from the network controller to sensor nodes before they are deployed.

### 3.3.2 Securing Data Storage

After obtaining  $\overline{f}_u(y)$  and before Phase  $i_s$  starts, sensor node  $u$  must derive the initial data encryption keys for Phases  $i_s, \dots, i_e$ . Following the notations used in Section 3.2, the initial key for Phase  $i$ , denoted as  $K_{u,i,0}$  is computed as the most significant  $l - r$  bits of  $\overline{f}_u(i)$ . Then,  $\overline{f}_u(y)$  is erased immediately.

Having derived the initial keys  $K_{u,i,0}$  ( $i = i_s, \dots, i_e$ ), the procedures for constructing data encryption keys and securing data storage is the same as Steps (2) and (3) of the EHB scheme (in Section 3.2), except that sensor node  $u$  generates and stores  $h(K_{u,i,0})$  before  $K_{u,i,0}$  is erased.

Following the example shown in Figure 1, Figure 2 illustrates the data storage procedure by showing how node 1

derives the keys for data encryption. After node 1 has derived  $\overline{f_1}(y)$ , it further derives initials keys for phases, i.e., (01010) for Phase 0, (11001) for Phase 1, and so forth. Then, during Phase 1, the first data item (with number 0) is encrypted with key (11001), the second item (with number 1) is encrypted with key  $h(11001)$ , and so forth.

### 3.3.3 Securing Data Retrieval

Let us consider a user who is authorized to query sensor data generated during Phase  $i$  ( $i \in \{i_s, \dots, i_e\}$ ). The procedure for the user to generate data decryption keys includes the following steps:

#### Data Decryption Polynomial Preloading

Before the user can access the sensor data he is authorized to, he is preloaded with a secret *data decryption polynomial*, denoted as  $\underline{f_i}(x)$ , by the network controller. Here,  $\underline{f_i}(x)$  is derived from the master polynomial  $f(x, y)$ ; specifically,

$$\underline{f_i}(x) = f(x, i) + \beta_i, \quad (2)$$

where  $\beta_i$  is an element of  $F_q$  randomly picked from  $\{0, \dots, 2^r - 1\}$ .

#### Data Retrieval and Decryption

Suppose the user has sent out his query and has received data response from sensor node  $u$ . The received data cannot be accessed immediately since they are encrypted. Specifically, each response data item has the following format

$$\langle \{D\}_{K_{u,i,j}}, i, j, t \rangle,$$

and node  $u$  also sends  $h(K_{u,i,0})$  to the user to facilitate him in deriving key  $K_{u,i,j}$ .

To figure out  $K_{u,i,j}$  and thus decrypt the data item, the user first evaluates the preloaded decryption polynomial  $\underline{f_i}(x)$  at  $x = u$  and obtains  $\underline{f_i}(u)$ . From  $\underline{f_i}(u)$ , the user further computes  $\underline{f_i}(u) - 2^r$  and  $\underline{f_i}(u) + 2^r$ . Since the above three numbers are all elements of  $F_q$ , each of them can be represented as  $l$  binary bits. Let us represent the most significant  $l - r$  bits of  $\underline{f_i}(u)$ ,  $\underline{f_i}(u) - 2^r$ , and  $\underline{f_i}(u) + 2^r$  as  $\gamma$ ,  $\gamma^-$  and  $\gamma^+$ , respectively. Then, we have the following theorem:

**Theorem 1**  $K_{u,i,0}$  must be the same as one of  $\gamma$ ,  $\gamma^-$ , or  $\gamma^+$ .

**Proof.** By comparing Eq. (1) and Eq. (2), we can see that both  $\overline{f_u}(i)$  and  $\underline{f_i}(u)$  are derived from  $f(u, i)$ ; their differences are determined by the added random numbers  $(\mu_v + \alpha_u + r_v)$  and  $\beta_i$ . Recall that both  $(\mu_v + \alpha_u + r_v)$  and  $\beta_i$  are in  $\{0, \dots, 2^r - 1\}$ , and  $K_{u,i,0}$  is the most significant  $l - r$  bits of  $\overline{f_u}(i)$ . So, there are following four cases:

- Case 1: Neither  $f(u, i) + (\mu_v + \alpha_u + r_v)$  or  $f(u, i) + \beta_i$  generates a carry from the least significant  $r$  bits to the most significant  $l - r$  bits. In this case, the most significant  $l - r$  bits of  $f(u, i)$ ,  $\overline{f_u}(i)$  and  $\underline{f_i}(u)$  are the same. Thus,  $K_{u,i,0} = \gamma$ .
- Case 2:  $f(u, i) + (\mu_v + \alpha_u + r_v)$  generates a carry from the least significant  $r$  bits to the most significant  $l - r$  bits, but  $f(u, i) + \beta_i$  does not. In this case, the most significant  $l - r$  bits of  $f(u, i) + (\mu_v + \alpha_u + r_v)$  is the same as those of  $f(u, i) + \beta_i + 2^r$ . Thus,  $K_{u,i,0} = \gamma^+$ .
- Case 3:  $f(u, i) + (\mu_v + \alpha_u + r_v)$  does not generate a carry from the least significant  $r$  bits to the most significant  $l - r$  bits, but  $f(u, i) + \beta_i$  does. In this case, the most significant  $l - r$  bits of  $f(u, i) + (\mu_v + \alpha_u + r_v)$  is the same as those of  $f(u, i) + \beta_i - 2^r$ . Thus,  $K_{u,i,0} = \gamma^-$ .
- Case 4: Both  $f(u, i) + (\mu_v + \alpha_u + r_v)$  and  $f(u, i) + \beta_i$  generate a carry from the least significant  $r$  bits to the most significant  $l - r$  bits. In this case, the most significant  $l - r$  bits of  $f(u, i)$ ,  $\overline{f_u}(i)$  and  $\underline{f_i}(u)$  are the same. Thus,  $K_{u,i,0} = \gamma$ .

According to the above theorem, the initial data encryption key of Phase  $i$  used by sensor node  $u$ , i.e.,  $K_{u,i,0}$  must be the same as one of  $\gamma$ ,  $\gamma^-$  or  $\gamma^+$ . Then,  $K_{u,i,0}$  can be found based on the following

$$h(K_{u,i,0}) = h(\gamma) \Rightarrow K_{u,i,0} = \gamma$$

$$h(K_{u,i,0}) = h(\gamma^-) \Rightarrow K_{u,i,0} = \gamma^-$$

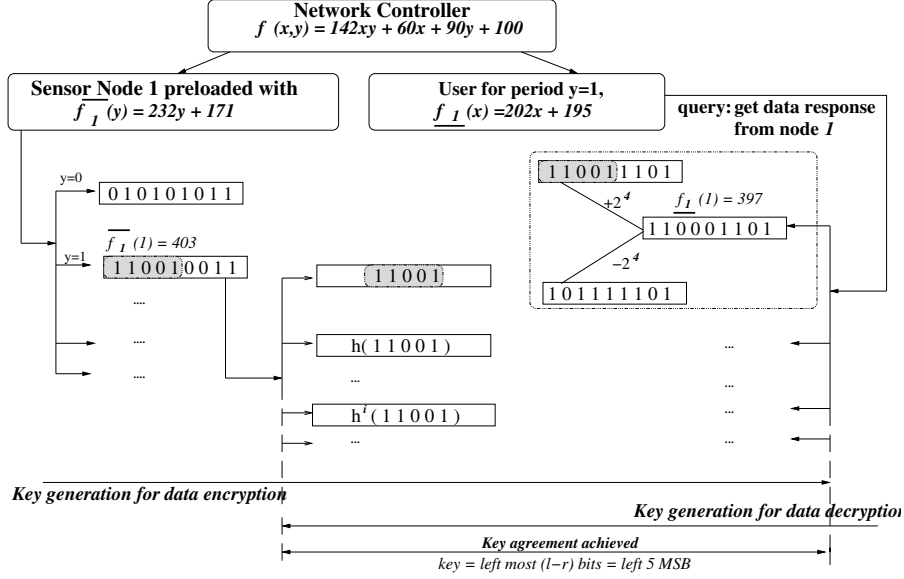
$$h(K_{u,i,0}) = h(\gamma^+) \Rightarrow K_{u,i,0} = \gamma^+$$

Once  $K_{u,i,0}$  is found, the key  $K_{u,i,j}$  can be computed by  $h^j(K_{u,i,0})$ ; hence, the data  $\{D\}_{K_{u,i,j}}$  can be decrypted.

Fig. 2 illustrates the above procedure of recovering the key to decrypt data.

### 3.3.4 Reducing Bandwidth Consumption

Every time when the data encryption keys should be refreshed, the network controller broadcasts  $w(x, y)$  to all sensor nodes. Let  $t$  be the degree of  $x$  and  $y$  in  $w(x, y)$ . Then,  $(t + 1)^2$  coefficients should be broadcast. It may still involve high communication overhead if  $t$  is large. Slight variation in the degree of polynomial greatly affects the amount of energy utilized to disseminate the perturbed shares. To reduce the communication overhead while not compromising the security properties of the scheme, we propose an optimization, in which the network controller broadcasts only  $(t + 1)$  coefficients instead of all  $(t + 1)^2$  coefficients. Using a relation defined below among coefficients, other coefficients can be reconstructed from the broadcasts  $(t + 1)$  coefficients. This optimization scheme reduces the requirement for broadcast bandwidth, still retaining the same features provided by the base scheme.



**Figure 2.** An Example of Generating Data Encryption Key at the sensor Node and Recovering the Key at the Receiver ( $r = 4$ )

Let

$$w(x, y) = \sum_{0 \leq i, j \leq t} A_{i,j} x^i y^j.$$

In addition, let the relation among the coefficients  $A_{i,j}$  be

$$A_{j,i} = h^j(A_{0,i}), \quad j = 0, \dots, t; i = 0, \dots, t.$$

and each  $A_{0,i}$  is an integer randomly picked from finite field  $F_q$ . The above defined relation among coefficients is known to all sensor nodes.

When the network controller needs to broadcast  $w(x, y)$ , it randomly generates only  $(t + 1)$  coefficients  $A_{0,0}, A_{0,1}, \dots, A_{0,t}$ , and broadcasts these coefficients to all sensor nodes. Based on the relation defined above, both network controller and sensor nodes can obtain the complete set of coefficients  $A_{i,j}$  ( $0 \leq i, j \leq t$ ). From the randomly constructed  $w(x, y)$ ,  $f(x, y)$  can be derived as follows:

$$f(x, y) = w(x, y) - p(x, y, v) - \mu_v,$$

where  $p(x, y, v)$  is defined earlier for the  $(v + 1)^t h$  dissemination (for  $v \in \{0, \dots, m - 1\}$ ), and  $\mu_v$  is randomly picked from  $\{0, \dots, 2^{r-1} - 1\}$ .

This optimization scheme for broadcast reduces  $1/(t + 1)$ -th times the energy consumed for per transmission of new seed polynomials. For  $t = 15$ , all the 256 (i.e.,  $(15 + 1) * (15 + 1)$ ) coefficients of  $w(x, y)$  has to be broadcast to all sensor nodes per update in original scheme. By using the optimization, the network controller needs to broadcast only 16 (i.e.,  $(15 + 1)$ ) coefficients. The communication bandwidth requirement is reduced by square root times the original requirement.

## 4. Security Analysis

In this section, we analyze the security properties of the proposed SHB, EHB and APB schemes. Following the attack model specified earlier, our analysis will focus on two types of attacks: (1) the attacker compromising a sensor node attempts to decrypt data stored earlier in the node; (2) the attacker attempts to decrypt data that it is not authorized to access.

### 4.1. Security Analysis for the SHB and EHB Schemes

When the SHB scheme is deployed, the attacker compromising a sensor node has negligibly low probability to decrypt data stored earlier in the node. This is because, it has negligibly low probability to derive any previously used key (which has been erased) from the current key due to the one-way property of secure hash functions employed. For the same reason, with the EHB scheme, the attacker compromising a sensor node also has negligibly low probability to decrypt data stored earlier in the node.

In SHB, any data retrieval and decryption must go through the network controller, and therefore, the attacker cannot decrypt data that it is not authorized to access. With EHB scheme, although users are given some keying information to enable them to derive keys for decrypting data they are authorized to, they have negligibly low probability to derive other keys based on the given keying information due to the one-way property of secure hash functions. For example, a user authorized to retrieve data generated during Phase  $i$  is given  $h(K_m | i)$  and thus can generate key  $h^j(h(K_m | i) | u)$  to decrypt the any data

item generated by any node during this phase. However, he cannot derive the keys used in any other Phase  $i'$ , i.e.,  $h^j(h(K_m | i') | u)$  since the probability for deriving  $h(K_m | i')$  from  $h(K_m | i)$  is negligibly low without knowing  $K_m$ .

## 4.2. Security Analysis for the APB Scheme

For the APB scheme, we consider two categories of attacks: attacks without collusions and attacks with collusions.

### 4.2.1 Attacks without collusions

Without collusions, the attacker who compromises a sensor node  $u$  may attempt to derive previously data encryption keys to decrypt data stored earlier. Based on the timing of the key to be derived, there are three cases:

- (i) To derive a previous key in the current phase: Due to the one-way property of secure hash functions, the probability of successful derivation is negligibly low.
- (ii) To derive a previous key that was generated based on a seed polynomial that has been updated: Due to the arbitrariness in picking seed polynomials, there is no connection between different seed polynomials. Therefore, the probability of successful derivation is also negligibly low.
- (iii) To derive a previous key in some previous phase, and the key to be derived and the current key are generated based on the same network-wide seed polynomial  $f(x, y)$ .

In this case, the attacker may launch brute-force attack to guess the key, and the complexity is  $\Omega(2^{L_k})$ , where  $L_k$  is the size of key. For  $L_k = 64$  bits, the complexity will be as high as  $2^{64}$ . Another approach is to break the seed polynomial of sensor node  $u$ , i.e.,  $\overline{f_u}(y)$  based on the initial keys of phases that still stored in node  $u$ , and thus can break the keys in previous phases based on  $\overline{f_u}(y)$ . However, according to the following theorem, the complexity of this attack is as high as  $\Omega(2^{(r-1)*(t+1)})$ .

**Theorem 2** Let  $\overline{f_u}(y) = \sum_{j=0}^t B_j y^j$ . The initial keys of  $n$  phases ( $i = 0, \dots, n-1$ ) are  $K_{u,i,0}$  ( $i = 0, \dots, n-1$ ), each is the most significant  $l-r$  bits of  $\overline{f_u}(i)$ . Suppose the attacker has know all  $K_{u,i,0}$  ( $i = 0, \dots, n-1$ ), the complexity to find out  $\overline{f_u}(y)$  is  $\Omega(2^{(r-1)*(t+1)})$ .

**Proof.** To find out  $\overline{f_u}(y)$ , the attacker needs to find out its  $(t+1)$  the coefficients  $B_0, B_1, \dots, B_t$ . Since the adversary knows  $n$  keys, it can obtain the following system of linear equations:

$$K_{u,i,0} * 2^r = \sum_{j=0}^t B_j (i)^j - R_i, \quad i = 0, \dots, n-1.$$

Here,  $R_i$  represents the least significant  $r$  bits of  $\overline{f_u}(i)$ , and thus  $K_{u,i,0} * 2^r = \overline{f_u}(i) - R_i$ .  $B_j$  ( $0 \leq j \leq t$ ) and  $R_k$  ( $1 \leq k \leq n$ ) are unknowns. Therefore the total number of unknowns are  $n+t+1$ , while the total number of linear equations is  $n$ . Since the number of linear equations are less than the number of unknowns, the unique solution of  $B_i$  cannot be found. The only way to solve  $\overline{f_u}(y)$  is to guess some variables  $B_j$  ( $0 \leq j \leq t$ ) or  $R_k$  ( $1 \leq k \leq n$ ). Suppose the adversary guesses  $n_1$  coefficients  $B_j$  and  $n_2$  random number  $R_k$ .  $n_1$  and  $n_2$  must satisfy  $n+t+1 - (n_1+n_2) \leq n$ . This is because the number of remaining unknowns cannot exceed the number of equations. Therefore, we have  $n_1+n_2 \geq t+1$ . Since  $R_k$  are of  $(r-1)$  bits, which are shorter than the coefficients  $B_j$ . The adversary must choose to guess  $t+1$  of  $R_k$ 's. Each  $R_k$  is picked from a set of  $2^{(r-1)}$  numbers. Since there is only a unique solution, the expected time complexity to guess the right answer is  $\Omega(2^{(r-1)*(t+1)})$ .

### 4.2.2 Attacks with collusions

If the attacker has compromised multiple sensor nodes, the information captured from all these nodes can be combined to derive the previous keys used by one of the compromised node  $u$ . Here, we consider the extreme scenario (most favoring the attacker). That is, we assume the attacker has captured  $\overline{f_{v_i}}(y)$  from  $n$  sensor nodes  $v_0, \dots, v_{n-1}$ . Based on the information, the attacker attempts to break  $f(u, y)$  and thus breaking any key used by sensor node  $u$ . However, as shown in the following theorem, the complexity of this attack is as high as  $\Omega(2^{(r-2)*(t+1)})$ .

**Theorem 3** Let  $f(x, y)$  be a network-wide seed polynomial, and the degree of  $x$  and  $y$  be  $t$ .  $\overline{f_{v_i}}(y)$  ( $v_0, \dots, v_{n-1}$ ) are  $n$  perturbed shares of  $f(x, y)$ ; i.e.,  $\overline{f_{v_i}}(y) = f(v_i, y) + \alpha_{v_i}$ , where  $\alpha_{v_i} \in \{0, \dots, 2^{r-1} - 1\}$ . Suppose the attacker has know all  $\overline{f_{v_i}}(y)$  ( $v_0, \dots, v_{n-1}$ ), given an arbitrary pair of  $u$  and  $t$ , the complexity to find out an arbitrary  $f(u, t)$  is  $\Omega(2^{(r-2)*(t+1)})$ .

**Proof.** Similar to the proof of Theorem 2.

**Theorem 4** Let  $\widetilde{p_u}(y, z)$  be a node  $u$ 's perturbing polynomial, and the degree of  $y$  and  $z$  be  $t$ .  $\overline{p_u}(y, v)$  ( $\forall v \in \{0, \dots, m-1\}$ ) are  $m$  perturbed shares of  $\widetilde{p_u}(y, z)$ ; i.e.,  $\overline{p_u}(y, v) = \widetilde{p_u}(y, v) - r_v$ , where  $r_v \in \{0, \dots, 2^{r-2} - 1\}$ . Suppose the attacker knows all  $\overline{p_u}(y, v)$  ( $\forall v \in \{0, \dots, m-1\}$ ), the complexity to find out  $\widetilde{p_u}(y, z)$  is  $\Omega(2^{(r-3)*(t+1)})$ .

**Proof.** Similar to the proof of Theorem 2.

## 5. Performance Evaluation

Here in this section we report the results for implementing the proposed adaptive polynomial-based (APB) scheme.

We have implemented the schema on top of the TinyOS platform, and simulated it in TOSSIM. We used the performance evaluation script for mica2 motes to calculate the storage, computation, and communication overhead.

In our implementation we deploy two types of nodes, sensor nodes, and user nodes with embedded server functionality. The system parameters  $l$  and  $r$  are set to be 64 bits and 16 bits, respectively. Hence, the size of each data encryption key is 48 bits. The degree of  $x$  and  $y$  in polynomials  $p(x, y)$  and  $f(x, y)$  is fixed at 15. The duration of each phase is 1 hour, and the interval for disseminating seed polynomials varies between 10 hours and 100 hours.

We measure the sensor mote’s CPU cycle time for one polynomial share evaluation (including conversion of polynomials to keys), one data message storage (including data encryption and hashing of key), one query response (includes retrieval, and sending of encrypted data and hashed key for the user mote query) and one seed polynomial receiving (includes receiving the seed polynomial from the network controller, evaluating the corresponding shares for this sensor node). At the user node, we measure the CPU cycles for one query sending and response data decryption.

The experimental results are reported in the following. As we can see, the overhead of the proposed APB scheme is low and affordable for current generation of network, in terms of storage, communication and computation overhead.

### 5.1. Storage Overhead

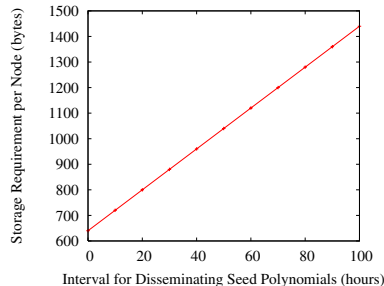
Nodes	ROM Size (bytes)	RAM Size (bytes)
SENSOR	23438 (code) + 128 (data)	720 ~ 1440
USER	25410	745

**Table 1.** Storage Overhead for Sensor and User Nodes

As shown in Table 1, the storage overhead experienced in our implementation at the user node is 745 bytes of RAM required to store the polynomials sent by the network controller for the data retrieval, and at the sensor node is between 720 (when the interval for disseminating seed polynomial is 10 hours) and 1440 (when the interval is 100 hours) bytes of RAM required for storing data encryption keys. The program code consumes 25.4 KB and 23.4 KB at the user and sensor nodes, respectively.

We also measure the storage requirement as the interval for disseminating seed polynomials vary, and the result is shown in Figure 3. In the figure, the y-axis stands for the total storage overhead required to store sufficient keys at the sensor node till it receives a new seed polynomial from the network controller. The x-axis represents the share update interval, which is varied between 10 hours and 100 hours. The figure shows that the storage overhead caused by key storage is directly proportional to the interval of disseminating seed polynomials. Smaller the update interval

time (seed polynomials are disseminated often, say every 10 hours), the storage required to store sufficient keys at the sensor node remains low (720 bytes). As the time interval increases (as we increase along the x-axis, seed polynomials are sent every 100 hours), the sensor node needs to store a large amount of keys incurring greater storage overhead (1440 bytes).

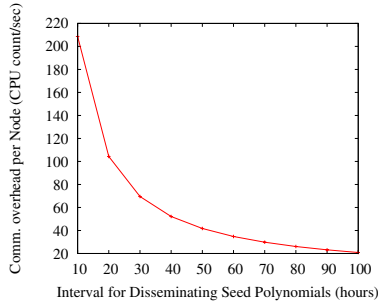


**Figure 3.** Storage Overhead vs. Interval for Dissemination Seed Polynomials

### 5.2. Communication Overhead

We measure communication overhead as the interval for disseminating seed polynomials vary, and the result is shown in Figure 4. In this figure, the y-axis represents the average communication overhead per second at each sensor node, for receiving seed polynomials from the network controller. As the seed polynomials are broadcast, the sensor network is flooded with the polynomials. So each sensor node not only experiences communication overhead for receiving one set of coefficients, but also encounters overhead for forwarding the coefficients along with receiving redundant sets of coefficients. So we have randomly simulated various topologies with randomly generated neighbors for calculating the average communication overhead at each sensor node. The overhead is measured in terms of CPU cycle counts experienced per second.

The graph shows that the communication overhead caused by disseminating and receiving seed polynomials is inversely proportional to the interval of dissemination. Smaller the update interval time (that is updates are sent often, say every 10 hours), the communication overhead experienced at the each sensor node rises greatly (approximately 208 CPU cycles/sec). As the time interval between the two dissemination increases (as we increase along the x-axis, the update is sent very rarely, say, every 100 hours), the sensor node incurs less communication overhead (approximately 21 CPU cycles/sec).



**Figure 4.** Communication Overhead vs. Interval for Disseminating Seed Polynomials

Functions	CPU Cycle Count	CPU Cycle Time (ms)
Sensor: one poly. evaluation and generation of initial keys	104,793.5	14.19
Sensor: one data encryption and key hashing	47,935.0	6.493
User: initialization	15,464.0	2.0945
User: one query and data retrieval	105,732.5	14.32

**Table 2.** Computational Overhead in Sensor and User Nodes

### 5.3. Computational Overhead

Each sensor node consumes 14.19 ms of CPU cycle time for generation of initial keys for phases from the received seed polynomials; 6.493 ms CPU cycle time is consumed for securing the storage of each data item. The user node initiation consumes 2.0945 ms CPU Cycle time, which involves initial setup with Server Node. Each query consumes 14.32 ms CPU cycle time.

## 6. Conclusion

In this paper, we proposed three schemes for securing distributed data storage and retrieval in sensor networks. All the schemes have the following properties: (i) Only authorized entities can access data stored in the sensor network; (ii) The schemes are resilient to a large number of sensor node compromises. The second and the third schemes do not involve any centralized entity except for a few initialization or renewal operations, and thus support secure, distributed data storage and retrieval. The third scheme further provides high scalability and flexibility, and hence is most suitable in real applications. The effectiveness and efficiency of the proposed schemes have also been verified through extensive analysis and TOSSIM-based simulations.

## References

- [1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, March 2002.
- [2] G. Wang, G. Cao, and T. La Porta, "A Bidding Protocol for Deploying Mobile Sensors," *IEEE International Conference on Network Protocols (ICNP)*, November 2003.
- [3] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *ACM International Conference on Mobile Computing and Networking (MOBICOM'02)*, pp. 148–159, September 2002.
- [4] G. Wang, G. Cao, T. La Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," *IEEE INFOCOM*, March 2005.
- [5] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar, "Spins: security protocols for sensor networks," in *Proceedings of ACM Mobile Computing and Networking (Mobicom'01)*, 2001, pp. 189–199.
- [6] L. Eschenauer and V. Gligor, "A Key-management Scheme for Distributed Sensor Networks," *The 9th ACM Conference on Computer and Communications Security*, pp. 41–47, November 2002.
- [7] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *The 10th ACM Conference on Computer and Communications Security*, 2003.
- [8] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical En-route Filtering of Injected False Data in Sensor Networks," *IEEE Infocom'04*, March 2004.
- [9] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbaugh, "Toward Resilient Security in Wireless Sensor Networks," *ACM MOBIOHOC*, May 2005.
- [10] G. Wang, W. Zhang, G. Cao, and T. La Porta, "On Supporting Distributed Collaboration in Sensor networks," *IEEE Military Communications Conference (MILCOM)*, October 2003.
- [11] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan and S. Shenker, "GHT: A Geographic Hash Table for Data-Centric Storage," *WSNA '02*, September 2002.
- [12] W. Zhang, G. Cao, and T. La Porta, "Data Dissemination with Ring-Based Index for Sensor Networks," *IEEE International Conference on Network Protocol (ICNP)*, November 2003.
- [13] J. Newsome and D. Song, "GEM: Graph Embedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information," *In Proc. SenSys'03, Los Angeles, California, USA*, November 2003.
- [14] B. Greenstein, S. Ratnasamy, S. Shenker, R. Govindan, and D. Estrin, "DIFS: a distributed index for features in sensor networks," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 333–349, 2003.
- [15] X. Li, Y. Kim, R. Govindan, W. Hong, "Multi-dimensional range queries in sensor networks," *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*, pp. 63–75, 2003.
- [16] A. Ghose, J. Grobklags and J. Chuang, "Resilient data-centric storage in wireless ad-hoc sensor networks," *Proceedings the 4th International Conference on Mobile Data Management (MDM'03)*, pp. 45–62, 2003.
- [17] P. Desnoyers, D. Ganesan, and P. Shenoy, "TSAR: A Two Tier Sensor Storage Architecture Using Interval Skip Graphs," *In Proc. SenSys'05, San Diego, California, USA*, November 2-4 2005.
- [18] Q. Fang, J. Gao, and L. Guibas, "Landmark-Based Information Storage and Retrieval in Sensor Networks," April 2006.