Paper ID#: 901523.PDF

Efficient Fine-grained Data Access Control in Wireless Sensor Networks

Qian Wang, Kui Ren, and Cong Wang
Illinois Institute of Technology
Email: {qwang,kren,cwang}@ece.iit.edu

Wenjing Lou
Worcester Polytechnic Institute
Email: wjlou@ece.wpi.edu

***Abstract*—Recent advances in distributed in-network data storage and access control have led to active research in efficient and robust data management in wireless sensor networks (WSNs). Although numerous schemes have been proposed this far, most of them do not provide enough attention towards exploiting user hierarchy and sensor heterogeneity, which is quite a practical issue especially when deploying WSNs in mission-critical application scenarios. In this paper, we propose an efficient secret-key cryptography-based (SKC) fine-grained data access control scheme for securing both distributed data storage and retrieval. In our design, secret keying information for data encryption and decryption are constructed based on the scheme of Blundo et al. [1] with information-theoretic security. To further enhance the security strength, we then propose an efficient user revocation scheme based on the idea of blinded Merkle hash tree construction. Extensive performance analysis shows that the proposed schemes are very efficient and practical for WSNs.**

I. Introduction

Wireless sensor network (WSN) has been an area of significant research in recent years [2]–[5]. In a typical WSN, a large number of sensor nodes can be easily deployed to various terrains of interest to sense the environment. Due to its flexibility and scalability,WSN has found its wide applications in both civilian and military domains. To accomplish the targeted application and fulfill its functionalities, a WSN usually generates a large amount of sensed data continuously over its lifetime. One of the biggest challenge then is how to implement secure data storage and retrieval in WSNs.

Data storage and access in WSNs mainly follow two approaches, namely, centralized and distributed approaches. In the centralized case, sensed data are collected from individual sensors and transmitted back to a central location, usually the sink, for storage and access. In the distributed approach, after a sensor node has generated some data, the node stores the data locally or at some designated nodes within the network, instead of immediately forwarding the data to a centralized location out of the network. The stored data later on can be accessed in distributed manner by the users of the WSN. For example, in "Unattended WSN" [2], data collection is not performed in (or near) real time and data should survive for a long enough time to be collected. "In-Situ Data Storage WSNs" [3] store the sensor readings at the generating sensor node (In-Situ); "Storage-Centric WSNs" [4] store historical data for the applications that need to mine sensor logs to analyze historical trends; In "Asynchronous WSN" [5], sensor nodes do not transmit the data to authorized devices in real-time, but store the data itself; In "Data Centric WSN" [6], relevant data are stored by name at the network nodes, and all data with the same general name will be stored at the same nodes.

To the best of our knowledge, distributed data storage and access security as a fairly new area receives limited attention this far. Previous research on WSN security issues has been focused on network communication security, such as key management, message authentication, secure time synchronization and localization, and intrusion detection [7]–[10]. This becomes a more severe issue given the trend that more and more distributed in-network data storage and access/retrieval schemes [2], [5], [11]–[13] are being proposed. Usually, the distributed data storage and access is achieved by storing the data locally or at some designated nodes in an encrypted format, so that only the authorized users with the appropriate keys can access them. Recently, Subramanian *et al.* [13] proposed a coarse-grained SKC-based data storage and retrieval scheme, where data storage access control is achieved by sharing the symmetric key with authorized users based on perturbed polynomial technique. However, neither data types nor user types are differentiated and this polynomial-based scheme provides not much better security strength than the classical polynomial approach [1].

In mission-critical application scenarios such as hospital and battlefield, various kinds of sensors may generate various types of sensitive data, which may belong to different security levels and are meant to be accessed only by selected types of users. One way to enable differentiated data access control is to encrypt different types of data with different keys. But sensors storing the same type of data will be using the same keys. Then, a user allowed to access more types of data will store more keys so that he will be able to perform all the corresponding decryption operations successfully. However, in this approach once a sensor is compromised, the corresponding keys used by this sensor can now be used to decrypt all sensor network data of the same type. Another way to further enhance the data access security can be the following: different sensors uses different keys to encrypt even the same type of data. Sensor or user compromises will reveal no key information for non-compromised entities. However, this approach incurs a very high key management overhead and complexity. The number of keys to be managed is now linear to the product of sensor network size (i.e., the total number of sensor nodes) and the total number of different data types.

To address these limitations, in this paper we first propose an SKC-based fine-grained data access control scheme for securing both distributed data storage and access in WSNs. In our scheme, data encryption and decryption polynomial shares are preloaded to sensor nodes and authorized users respectively by the network controller. To ensure fine-grained data access control, we assign each sensor a secret data access key associated with the type of data it stores, so each sensor can encrypt the data with a unique key not only to the sensor itself but also to the data type. Meanwhile, user hierarchy is achieved by assigning different number of access keys to the users based on their security levels. We show through detailed analysis that these schemes are very effective and efficient.

Second, we propose two polynomial-based user revocation schemes: a basic scheme followed by a blinded Merkle hash tree-based construction. In practice, an efficient user revocation mechanism is essential to the data access security when user compromises has been detected or some users left the network. Through extensive analytical study, we show that the Merkle hash tree-based revocation scheme can greatly reduce the computational overhead and communication overhead during the revocation process compared to the basic scheme.

The rest of the paper is organized as follows. Section II introduces the system model, attack model and our design goal. The detailed description of our SKC-based fine-grained data access control scheme is provided in Section III. In Section IV, we discuss the further enhancements for handling user dynamics. Section V and VI gives the security analysis and performance evaluations, respectively. Finally, we conclude in VII.

II. Problem Statement

A. System Model

In this paper, we consider a wireless sensor network with a large number of sensor nodes, each of which has a unique ID. These nodes are equipped with sufficient capacity to store the sensed data locally in a distributed manner for a certain period. We follow the same assumptions adopted in [13]. That is, sensor nodes maintain loose time synchronization, and the lifetime of the network can be divided into phases, each having the same time duration.

We observe that in a mission-critical application such as battle field, the users of the WSN may inherently form a hierarchy regarding their accessibility to various data types based on their security levels. For example, in a battle field, a general may be able to access all types of data, while a major may only be entitled to access a subset of data and a soldier may access even less types of data. On the other hand, a tank may be entitled to have the same accessibility as a major does. In other words, we can divide the network users into different security classes, i.e., a number of disjoint sets each including a group of users with the same security clearance. In our system

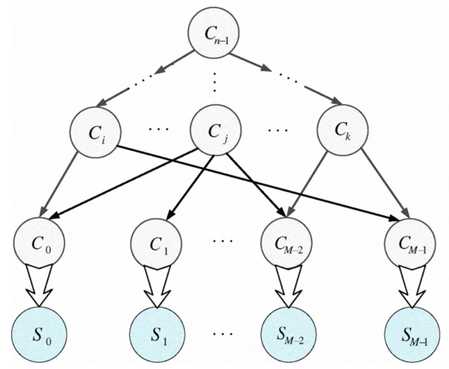

Fig. 1. Data access control hierarchy.

model, each system user is assigned a unique user ID and a secret security class ID . Assume $C_0, C_1, \ldots, C_{n-1}$ are n disjointed security classes. Let $\geq$ denote a binary partially ordered relationship in a user set $C = \{C_0, C_1, \ldots, C_{n-1}\}$. In the partially ordered set (*poset*, C, $\geq$), $C_i \geq C_j$ denotes that the security class C_i have a security clearance higher than or equal to the security class C_j. As for our scheme, a higher security clearance simply means that users can access more types of data. The access control hierarchy is presented in Fig. 1. In the network, sensors are divided into M different groups $S_0, S_1, \ldots, S_{M-1}$ that each group of sensors stores a certain type of data. On the other hand, network users are divided into different security classes based on their access privileges. Note $C_0, \ldots, C_{M-1}$ are the user security classes that have the lowest security clearance, i.e., the users in these classes can access only one type of data.

B. Adversary Model

The adversary's main objective is to steal data stored in the network. More specifically, the adversary has the following capabilities: (i) Individual sensors are not reliable since they can be compromised due to lack of tamper-proof hardware. Once a sensor is compromised, the adversary can read its memory and get all information stored there; (ii) Legitimate network users can also be compromised, the adversary can make use of the key materials held by the compromised users to obtain data that he is not authorized to access. Our design aims to reduce the damage caused by compromise and colluding attacks to the minimum.

C. Design Goal

Data access security in WSNs should ensure that sensor network data can only be accessed by authorized users. More specifically, we have the following goals: (i) Fine-grained differentiated user access control: By exploiting user hierarchy and sensor heterogeneity, we can ensure that different types of sensitive data can be accessed only by selected types of users based on their security clearance; (ii) Resilience against sensor or user compromise attacks: To decrease the gain on compromising individual sensors; (iii) Lightweight: The design for data access control mechanisms should be lightweight as always in order to fit into the inherent resource-limited nature of WSNs.

D. Notation

Now we list some notation used in presenting the scheme:

- q, l: q is a large prime number, l is the minimal integer such that $2^l > q$. The element in field F_q can be represented by l bits.
- $f_j(x, y)$ $(j \in \{0, \ldots, d-1\})$: a bivariate polynomial with degree at most t for both x and y, and it provides the basis for generating data encryption and decryption keys.
- SID_i: the unique ID for sensor node i. For a regular sensor node, we usually denote SID by v for simplicity.
- GID_m $(m \in \{0, \ldots, M-1\})$: the secret group ID for sensors that stores the mth type of data.
- K_m $(m \in \{0, \ldots, M-1\})$: the secret data access key which provides accessibility for the mth type of data.
- UID_i, CID_i: UID_i denotes the secret user ID for user i, CID_i denotes the secret ID for user security class i.

III. A SKC-Based Fine-Grained Data Access Control Scheme

In WSNs, symmetric key cryptography (SKC) based data access control is achieved by storing the data in encrypted format on the sensor nodes and sharing the appropriate symmetric keys with authorized users. However, traditional SKC-based schemes provide no support on fine-grained data access control due to the key management complexity. In this section, we propose to exploit both sensor heterogeneity and user hierarchy for designing a lightweight SKC-based fine-grained data access control scheme.

A. System Initialization

1) Sensor Initialization: Before a sensor node is deployed, the network controller assigns it a unique SID and a secret GID_m $(m \in \{0, \ldots, M-1\})$ based on the type of data it will store. Without loss of generality, we denote SID by v. Then the network controller arbitrarily constructs d univariate polynomials

$$\overline{f_{v,j}}(y) \quad = \quad f_j(v, y) + K_m, \quad j = 0, \ldots, d-1,$$

where K_m is the secret data access key which provides accessibility for the mth type of data.

2) User Initialization: Before users enter the network, the network controller assigns each of them a unique UID and a secret CID based on its security level. Moreover, the network controller will also distribute users the secret access keys accordingly, i.e., users will be equipped with different number of data access keys (K_ms) according to their security levels.

B. Sensor Data Storage

After obtaining $\overline{f_{v,j}}(y)$ $(j = 0, \ldots, d-1)$, sensor node v can derive the data encryption keys for different phases. Assume the lifetime of the network is divided into N phases. The data storage process is as follows:

(1) At the beginning of phase i $(i \in \{0, \ldots, N-1\})$, for each $\overline{f_{v,j}}(y)$ $(j = 0, \ldots, d-1)$, node v computes:

$$\overline{f_{v,j}}(i) \quad = \quad f_j(v, i) + K_m,$$

where $m \in \{0, \ldots, M-1\}$. The key segment denoted as $K_{v,i,j}$ for phase i is computed as $\overline{f_{v,j}}(i)$ **mod** q.

(2) A concatenation of these d key segments, denoted as $K_{v,i} = (K_{v,i,0}|\cdots|K_{v,i,d-1})$, is used as the data encryption key during phase i. All the data items generated in phase i are encrypted with the current encryption key $K_{v,i}$.

C. User Data Access Control

Assume an authorized user u wants to access data generated during phase i, it is preloaded with d data decryption polynomial $\underline{f_{i,j}}(x)$ by the network controller:

$$\underline{f_{i,j}}(x) \quad = \quad f_j(x, i), \quad j = 0, \ldots, d-1,$$

Based on his security level, the user can access some types of data using $\underline{f_{i,j}}(x)$ $(j = 0, \ldots, d-1)$ in conjunction with the data access keys he holds. Suppose the user has sent out his query and has received data response $< \{D\}_{K_{v,i}}, i, m >$ from sensor node v, where $\{D\}_{K_{v,i}}$ is the data item encrypted with the symmetric key $K_{v,i}$. i and m denote the identifications of the phase and data type, respectively. To figure out $K_{v,i}$, the user first evaluates each $\underline{f_{i,j}}(x)$ $(j = 0, \ldots, d-1)$ at $x = v$ to obtains $\underline{f_{i,j}}(v)$. Because v belongs to sensor group m $(m \in \{0, \ldots, M-1\})$, the user further computes $\underline{f_{i,j}}(v) + K_m$. The correct data decryption key must be $K_{v,i,0}|\cdots|K_{v,i,d-1}$. Then $\{D\}_{K_{v,i}}$ can be decrypted.

Discussion. It is easy to see that in our data access control scheme, each sensor encrypts the data with a different key, which is unique not only to the sensor itself but also to the data type. That is, different sensors uses different keys to encrypt even the same type of data, respectively. Compromising a sensor will only lead to the comprise of the data stored on this sensor; Compromising a user, on the other hand, will only lead to the compromise of the types of the data accessible by this user.

IV. Handling of User Dynamics and Insider Attacks

A. Ensuring User Dynamics

In our distributed data storage and retrieval system, the locally stored data are encrypted with symmetrical keys that are generated based on phases. Meanwhile, only authorized users equipped the decryption keys (based on phases) can access the data stored in the network. Consider a user who is detected compromised and should be revoked as soon as possible, the keying information he holds may be used by unauthorized entities to access the sensitive data or by adversaries to launch malicious attacks. Thus, to enhance the system security, this user must be dynamically removed from the network instantly. In the SKC-based data storage and access paradigm, to achieve a successful revocation operation, the network controller also needs to refresh the keying information stored in related sensor nodes. This guarantees that (i) a passive adversary who knows a contiguous subset of old encryption keys cannot discover subsequent encryption keys; (ii) a passive adversary who knows a contiguous subset of encryption keys cannot discover preceding encryption keys. Thus, both forward and backward secrecy are achieved. Actually during the revocation process sensors will re-encrypt all the data items using the updated encryption keys, to discover the preceding encryption keys will provide no advantage for the adversary to access the encrypted data. Moreover, since the user to be revoked belongs to certain security class (e.g., C_i), the network controller should not only refresh the keying information related to C_i, but also update the information among the related predecessors and successors on the connected path (see Fig. 1).

As a straightforward approach to the rekeying problem, the network controller may unicast the new key materials to each user individually. However, the communication overhead increase rapidly as the number of users increases. On the other hand, to revoke multiple users, the network controller has to repeat the process of rekeying for each revoked user. Hence, the cost of rekeying is high. Thus, an efficient user revocation scheme that readily accommodates the characteristics of our SKC-based data access control service is highly desirable.

1) Basic Scheme: Recently, many hierarchical key management schemes [14]–[16] have been proposed. All these schemes seek to provide a trade-off between the number of keys maintained by users and the time required for rekeying due to revocation of multiple users. In this paper, we consider a polynomial-based key management scheme [14] as the basis to construct our protocols. Assume a user (with UID_l) in security class C_i holds p ($p \leq M$) secret data access keys. Without loss of generality, let $K_1, \ldots, K_p$ denote the p secret access keys and $GID_1, \ldots, GID_p$ denote the corresponding secret sensor group ids. To revoke this user, the network controller will act as follows:

***Step 1**: Updating encryption keys* The network controller randomly selects a number γ from F_q and constructs

$$\overline{q_1}(x) = \prod_{i=1}^{p}(x - h(GID_i||z)) + \gamma.$$

Then, $(z, \overline{q_1}(x))$ are broadcasted to the sensor nodes. Note that only sensors in groups with $GID \in \{GID_1, \ldots, GID_p\}$ can compute γ as $\gamma = q_1(h(GID||z))$. After deriving γ, the sensor can update its data encryption keys by computing $\overline{f_{v,j}(i)} = f_j(v,i) + K + \gamma$ for $j = 0, \ldots, d-1$, where $K \in \{K_1, \ldots, K_p\}$. Then, node v re-encrypts all the data items using the updated $K_{v,i}$. In fact, in this process, K is updated by $K = K+\gamma$. Note that in our scheme the revocation messages are broadcasted and we have assumed that these revocation messages can arrive at the destination sensor nodes reliably and correctly. If a sensor is compromised, the GID for that group should also be updated. The detection of sensor compromises is orthogonal to this work.

***Step 2**: Updating secret class ID:* The network controller will select a random number μ form F_q and secretly distributes it to C_i's users. Similarly, the network controller constructs:

$$\overline{q_2}(x) = \prod_{i \in \Omega, i \neq l}(x - h(UID_i||z^{'})) + \mu,$$

and publicizes $(z^{'}, \overline{q_2}(x))$, where Ω denotes the serial number of users in security class C_i. Note that the term $x - h(UID_l||z^{'})$ is not included in the above polynomial. Thus any user in security class C_i except for the user to be revoked can obtain the μ by substituting x with $h(UID||z^{'})$, where UID denotes the user's secret ID. Then the user class ID for C_i is updated as $CID_i = CID_i + \mu$.

***Step 3**: Updating decryption keys:* For the remaining users (including users in higher security level classes) who hold any key in $\{K_1, \ldots, K_p\}$ should update their corresponding key information. Assume that w security classes are affected and required to update their key information. Without loss of generality, let $CID_1, \ldots, CID_w$ denote the ids of these secret classes. Note that we use the serial numbers $1, \ldots, w$ for simplicity, actually the affected classes include both low level and high level classes. The network controller constructs

$$\overline{q_3}(x) = \prod_{i=1}^{w}(x - h(CID_i||z^{''})) + \gamma,$$

and publicizes $(z^{''}, \overline{q_3}(x))$. By plugging $h(CID_i||z^{''})$ ($i \in \{1, \ldots, w\}$) into $\overline{q_3}(x)$, the users in these classes can obtain γ in order to update the access key by computing $K + \gamma$ ($K \in \{K_1, \ldots, K_p\}$). Therefore, to access the data stored by sensors in group $GID_1, \ldots, GID_p$, the legitimate users can compute $K_{v,i}$ using these new access keys. Note that $z, z^{'}, z^{''}$ are randomly chosen from F_q and made public for each revocation process.

To revoke multiple users, similarly, the network controller will construct $\overline{q_1}(x)$ and update the data encryption keys held by the related sensor groups; In the second step, $\overline{q_2}(x)$ is constructed by excluding the terms $x - h(UID_{l_1}||z^{'}), x - h(UID_{l_2}||z^{'}), \cdots$. Thus, all the remaining users can update their secret class IDs; Finally, $\overline{q_3}(x)$ is generated and broadcasted, so all the users in the affected classes can update their secret data access information.

Discussion. In practice, note that (i) once a user is detected compromised, it will be revoked as soon as possible. Thus, during each revocation operation, only a few users (e.g., 1 or 2) are to be revoked; (ii) the number of data types and the number of user security classes are small, so the main overhead exists in *step 2*, i.e., the distribution and computation of $\overline{q_2}(x)$. It is obvious that the basic scheme can meet our design goal and work well even when multiple users are required to revoked at one time. However, the basic revocation scheme is not efficient when only a few users are to be revoked. This is because in this case the degree of $\overline{q_2}(x)$ is very large, the resultant polynomial computation and communication complexity are high. This situation becomes even worse when the system has a large amount of legitimate users.

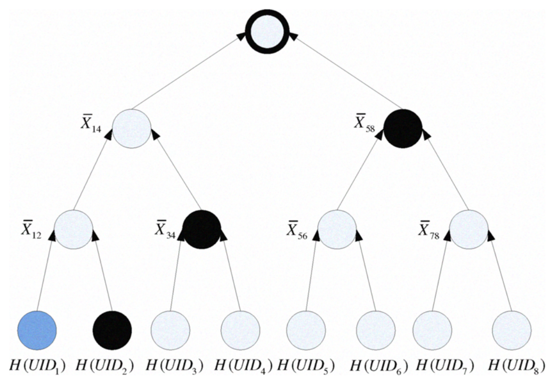

Fig. 2. A Merkle-hash tree constructed from the UIDs of the users.

2) A Blinded Merkle Hash Tree-based Scheme: To address the limitations in the basic scheme, we propose an efficient user revocation scheme based on the Merkle hash tree [17]. A Merkle Tree is a construction to build secure authentication schemes from hash functions. The leaves in the tree are hashes of the authentic data values. Nodes further up in the tree are the hashes of their respective children.

Suppose there are N users in security class C_i. Let the secret user IDs be $UID_1, \ldots, UID_N$. Our Merkle hash tree is constructed in a bottom-up fashion using the hash of the UIDs as the leaf nodes. A non-leaf node in the trees is a hash of its two child nodes, recursively until the root node X_{1N} is generated. Fig. 2 depicts an example where $N = 8$. In the construction,

$$\begin{aligned}
X_{12} &= H(H(UID_1)||H(UID_2)), \overline{X}_{12} = H(X_{12}),\\
X_{34} &= H(H(UID_3)||H(UID_4)), \overline{X}_{34} = H(X_{34}),\\
X_{56} &= H(H(UID_5)||H(UID_6)), \overline{X}_{56} = H(X_{56}),\\
X_{78} &= H(H(UID_7)||H(UID_8)), \overline{X}_{78} = H(X_{78}),\\
X_{14} &= H(\overline{X}_{12}||\overline{X}_{34}), \overline{X}_{14} = H(X_{14}),\\
X_{58} &= H(\overline{X}_{56}||\overline{X}_{78}), \overline{X}_{58} = H(X_{58}),\\
X_{18} &= H(\overline{X}_{14}||\overline{X}_{58}), \overline{X}_{18} = H(X_{18}),
\end{aligned}$$

where H is a collision-resistant hash function. Different from the applications of Merkle hash tree in authentication schemes, in our scheme, each user is preloaded with a small amount of auxiliary information during the registration process. The auxiliary values are the nodes sibling to the nodes UID_i to the root $\overline{X}_{iN}$. As an example, user 1 with UID_1 is preloaded with $H(UID_2), \overline{X}_{34}$ and $\overline{X}_{58}$. Suppose user 1 is to be revoked, we consider *step 2* in the revocation process. To revoke user 1, the network controller should update the secret information held by the remaining users. Thus, a polynomial $\overline{q_2}(x) = (x - UID_2)(x - X_{34})(x - X_{58}) + \mu$ can be constructed and broadcasted by the network controller. User 2 can obtain μ by substituting x with his own UID_2. Users 3 and 4 can obtain μ by first computing X_{34} and then plugging it to $\overline{q_2}(x)$. Users 5, 6, 7 and 8 can also easily derive μ by computing X_{58} using their own auxiliary information and plugging it into $\overline{q_2}(x)$. On the other hand, user 1 only has the hash (blinded) values of $UID_2, \overline{X}_{34}$ and $\overline{X}_{58}$, so it is impossible for him to derive the secret μ. It is easy to see that, compared to the basic scheme, our blinded Merkle hash tree-based revocation scheme can revoke the users more efficiently. In the example, the degree of $\overline{q_2}(x)$ is greatly reduced (from 7 to 3) compared with the basic scheme, the complexity of polynomial generation and communication are reduced and the efficiency of secret computation and derivation are improved.

Now we consider when multiple users are to be revoked. For simplicity, we first investigate the two-user revocation scenarios:

(1) Users 1 and 2: To revoke users 1 and 2, the network controller will construct $\overline{q_2}(x) = (x - X_{34})(x - X_{58}) + \mu$.
(2) Users 1 and 3: To revoke users 1 and 3, the network controller will construct $\overline{q_2}(x) = (x - UID_2)(x - UID_4)(x - X_{58}) + \mu$.
(3) Users 1 and 5: To revoke users 1 and 5, the network controller will construct $\overline{q_2}(x) = (x - UID_2)(x - X_{34})(x - UID_6)(x - X_{78}) + \mu$.

It can be seen that (i) if the two users are leaf nodes of the second level non-leaf nodes, the degree of $\overline{q_2}(x)$ is only 2; (ii) if the two users are leaf nodes of the third level non-leaf nodes, the degree of $\overline{q_2}(x)$ is 3; (iii) if the two users are leaf nodes of the fourth level non-leaf nodes, the degree of $\overline{q_2}(x)$ increases to 4, this is actually the worst case. Similarly, we can further analyze the scenario when more than two users are to be revoked.

Next, we consider the scenario when new users join the network. Assume that each security class always maintains a fixed number of users, thus a determinate Merkle hash tree is constructed for each security class in the user initialization phase. When users want to join the network, there are two possible cases: (i) The security class that the user wants to join is full, e.g, all the legitimate IDs in this class are used up. In this case, the user will wait until some existing users finish their jobs and leave the network. Then network controller will update the secret information (user revocation) for the remaining users and assign an available UID and updated CID to the new user. Finally, the updated data decryption information are encrypted with CID and sent to the new user; (ii) The security class has unoccupied UIDs. In this case, the network controller simply assigns a free UID and the old CID to the new user. Then, the data decryption information are encrypted with CID and sent to the new user.

V. Security Analysis

In this section, we analyze the security of our proposed schemes. After the adversary has compromised some sensor nodes and captured the data encryption polynomials preloaded to these nodes during a certain period, the adversary expect to attack the system based on these polynomial shares of $f_j(x, y)$. In additional, some legitimate users may also be compromised (or colluding together), thus the adversary or colluding users can initiate attacks based on the captured data decryption polynomials. The security proof in [1] ensures that our scheme is unconditionally secure and t-collusion resistant. That is, up to t colluding sensors reveal no information on $f_j(x, y)$. Similarly, by compromising decryption polynomials of less than t different phases reveal no information on $f_j(x, y)$. To improve the security level, recently, two perturbed polynomial-based schemes are being proposed: blinding polynomial shares using random numbers [13] and blinding polynomial shares using random polynomials [18]. However, our analysis shows that the perturbation technique of using random numbers still has a threshold value if some users are compromised or colluding together (due to space limitation, we omit the vulnerability analysis). In addition, the heuristic security arguments given for the scheme using random polynomials do not hold [19], so these schemes provide no better performance than the previous information-theoretic secure schemes.

VI. Performance Analysis

In this section we analyze the performance of our final scheme. The system parameters l is set to be 10. The number of master polynomial $f_j(x, y)$ used in our application is $d = 8$. Thus, the size of each data encryption key is 80 bits. The degrees of x and y in all polynomial shares and perturbation polynomials are set to $t = 80$. We set $M = 10$ in our analysis.

A. Performance Analysis of Data Storage and Access Scheme

Computational Cost: Given the d polynomial shares $\overline{f_{v,j}}(y)$ $(j = 0, \ldots, d-1)$, a sensor node v can compute data encryption keys for different phases over its lifetime. Let the degree of $\overline{f_{v,j}}(y)$ be t. The coefficients in the polynomial are elements of F_q. Let us consider the computational cost for one data message storage and retrieval. To generate data encryption keys, sensor node v evaluates the ID of the current phase at $\overline{f_{v,j}}(y)$ for $j = 0, \ldots, d-1$. Note that the points at which the polynomials are evaluated are phase IDs chosen from the same finite field F_q. For a system user, it evaluates the id of node v at its polynomial share $\overline{f_{i,j}}(x)$ for $j = 0, \ldots, d-1$. Thus, the computational cost is d polynomial evaluations with polynomial degree of t on both sensor and user side.

Communication Cost: The data retrieval process between a user and a sensor node involves totally two messages. The first message sent from the user to the sensor is a data query message. Upon receiving the query message, the sensor sends the response to the user. The response has the format $\langle \{D\}_{K_{v,i}}, i, m \rangle$, where $\{D\}_{K_{v,i}}$ is the required data item encrypted with the symmetric key $K_{v,i}$. i and m are small integers that denote the identifications of the phase and data type, respectively.

Storage Cost: In the initialization phase, a regular sensor node v only needs to store its data encryption polynomial shares $\overline{f_{v,j}}(y)$ $(j = 0, \ldots, d-1)$, i.e., $d * (t+1)$ coefficients. In addition to storing data decryption polynomial shares $\overline{f_{i,j}}(x)$ for $j = 0, \ldots, d-1$, a user stores a number of secret access key K_ms based on its security level, where K_ms are elements of F_q. As an example, a user in the security class with the highest clearance is equipped with M access keys, then the storage overhead is $M * l$ bits. Hence, even a user has to store a maximum of M access keys and its resources are as scare as sensor, the storage overhead is still not a concern.

TABLE I
Average Degree of Revocation Polynomial $q_2(x)$

Scheme			Number of users in a security class				
			64	128	256	512	1024
Merkle	$r = 1$	$O(\log_2 N)$	6.00	7.00	8.00	9.00	10.00
	$r = 2$	$O(\log_2^2 N)$	9.10	11.06	13.03	15.18	17.01
Basic	$r = 1$	$O(N)$	63.00	127.00	255.00	511.00	1023.00
	$r = 2$	$O(N)$	62.00	126.00	254.00	510.00	1022.00

B. Performance Analysis of Dynamic User Revocation Scheme

We consider the scenario that users to be revoked are in the same security class, the revocation of users from different security classes can follow the similar analysis. During the user revocation process, three revocation polynomials $\overline{q_1}(x), \overline{q_2}(x), \overline{q_3}(x)$ are broadcasted. The degree of these polynomials greatly affects the communication overhead (energy utilized to disseminate and receive the polynomial) and computational overhead (key computation and derivation). We establish two metrics to evaluate the performance of the proposed revocation scheme: the *average degree of revocation polynomial* and the storage overhead. The average degree of revocation polynomial is the average of all possible degrees of the constructed revocation polynomial when a fixed number of users is to be revoked from a security class. The storage overhead is defined as the total number of hash values prestored in the user node.

Now let us consider the cost for revoking a small number of users in a security class. Without loss of generality, we assume security class C has N legitimate users. Each user of C possesses p $(p \leq M)$ data access keys $K_0, \ldots, K_{p-1}$. The revocation process totally involves three messages (see Section IV-A). The first message sent from the network controller is $\overline{q_1}(x) = \prod_{i=1}^{p}(x - h(GID_i||z)) + \gamma$ of degree p, which is used to update the data encryption keys. The second message is $\overline{q_2}(x) = \prod_{i \in \Omega, i \neq l}(x - h(UID_i||z'')) + \mu$, which is used to update the secret class ID of C. The third message is $\overline{q_3}(x) = \prod_{i=1}^{w}(x - h(CID_i||z'')) + \gamma$, which is used to update the data decryption keys of the users in all the affected security class. It is easy to see that messages $\overline{q_1}(x)$ and $\overline{q_3}(x)$ has degrees t and w, respectively. That is, $(p+1) + (w+1)$ coefficients should be broadcasted no matter how many users are revoked from C. However, the degree of $\overline{q_2}(x)$ is dependent on the distribution of the users to be revoked in the Merkle hash tree.

Proposition 1: Assume a Merkle hash tree constructed for a security class C uses $N = 2^n$ UIDs as the leaf nodes. To revoke k $(0 \leq k \leq N)$ users at one time, the degree of the the revocation polynomial $\overline{q_2}(x)$ is at most $N/2$.

The proof is quite easy to obtained from the analysis in Section IV-A. Here we omit the proof due to space limitations. Proposition 1 gives the degree bound of the revocation

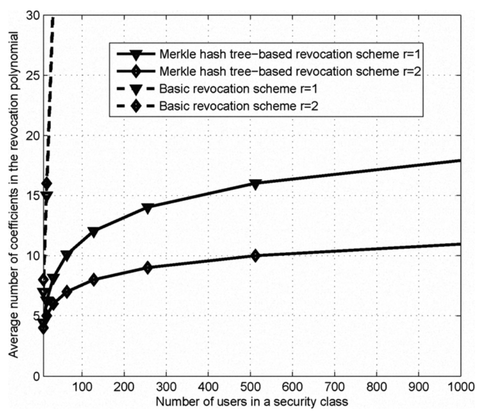

Fig. 3. Average number of coefficients in the revocation polynomial versus the number of users in a security class.

polynomial $\overline{q_2}(x)$. It implies that no matter how many users are to be revoked, the Merkle hash tree-based revocation scheme is always more efficient than the basic revocation scheme. Let r denote the number of users to be revoked during each revocation process. As shown in Table I, the average degree of revocation polynomial $\overline{q_2}(x)$ can be reduced significantly by using Merkle hash tree-based revocation scheme. Note that the computational cost and the communication cost are closely related to the *average degree of revocation polynomial*. Consider $r = 1$ and $N = 64$, compared with the basic scheme, the Merkle hash tree-based scheme can reduce the average degree from 63 to 6. Hence, the average communication cost can be reduced from $10*64$ bits to $10*7$ bits. At the user side, the number of modular multiplications associated with polynomial evaluation is also reduced. Fig. 3 shows the number of coefficients in $\overline{q_2}(x)$ as the number of users in a security class increases. In the Merkle hash tree-based revocation scheme, each user should be preloaded with some auxiliary information in a Merkle hash tree. The overall storage cost for a user in a security class of size N is equivalent to $\log_2 N$ hash values. Assume we use SHA-1 as the hash algorithm. Hence, even if a user is in a security class of $N = 1024$ users and its resources are as scarce as those of a sensor node, the storage overhead ($\log_2 N = 10$ hash values) is not a concern.

Discussion. In the analysis of communication overhead, we have implicitly assumed that the overhead mainly comes from the degree of the revocation polynomial, i.e., the number of coefficients. In practice, the transmitted message (the coefficients of polynomial) has to travel several hops before it reaches the destination sensor node. Thus, the communication overhead is also proportional to the number of hops needed to transmit the revocation message. To reduce the communication overhead incurred by multi-hop transmission, we may use the GPSR-based scheme for our purpose [20].

VII. Conclusion

In this paper, we propose an efficient SKC-based fine-grained data access control scheme for securing both distributed data storage and retrieval. In our design, keying information for data encryption and decryption are generated based on the Blundo's scheme [1] with information-theoretic security. To further ensure the data access security, we then propose an efficient user revocation scheme based on the idea of blinded Merkle hash tree. Through detailed performance analysis, we show that the proposed schemes are highly secure and efficient, thus can be implemented in the current generation of sensor networks.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CNS-0831963, CNS-0716306, CNS-0831628 and CNS-0746977.

References

[1] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," *Lecture Notes in Computer Science*, vol. 740, p. 471C48, 1993.
[2] R. D. Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "Catch me (if you can): Data survival in unattended sensor networks," in *IEEE PerCom'08*, Match 2008.
[3] D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos, A. Mitra, A. Banerjee, and W. Najjar, "Towards in-situ data storage in sensor databases," in *Proc. of PCI'05, LNCS*, 2005.
[4] Y. Diao, D. Ganesan, G. Mathur, and P. Shenoy, "Rethinking data management for storagecentric sensor networks," in *CIDR'07.*
[5] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, "Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks," *Ad Hoc Networks, Elsevier*, vol. 5, no. 7, pp. 1073–1089, 2007.
[6] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with GHT, a geographic hash table," *Mob. Netw. Appl.*, vol. 8, no. 4, pp. 427–442, 2003.
[7] F. Ye, H. Luo, S. Lu, L. Zhang, and S. Member, "Statistical en-route filtering of injected false data in sensor networks," in *IEEE INFOCOM'04.*
[8] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "Spins: Security protocols for sensor networks," *ACM Wireless Networks*, Sep. 2002.
[9] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," in *ACM Workshop on Security in Ad Hoc and Sensor Networks*, 2003.
[10] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Lhap: A lightweight hop-by-hop authentication protocol for ad-hoc networks," in *ICDCSW'03.*
[11] D. Ma and G. Tsudik, "Forward-secure sequential aggregate authentication," in *IEEE Security and Privacy'07*, Oakland, May 2007.
[12] W. Zhang, H. Song, S. Zhu, and G. Cao, "Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks," in *ACM MOBIHOC*, USA, May 2005.
[13] N. Subramanian, C. Yang, and W. Zhang, "Securing distributed data storage and retrieval in sensor networks," in *IEEE PerCom'07.*
[14] X. Zou, Y.-S. Dai, and E. Bertino, "A practical and flexible key management mechanism for trusted collaborative computing," in *IEEE INFOCOM 2008.*
[15] M. J. Atallah, K. B. Frikken, and M. Blanton, "Dynamic and efficient key management for access hierarchies," in *CCS '05.*
[16] M. Ramkumar and N. Memon, "An efficient key predistribution scheme for ad hoc network security," *IEEE JSAC*, 2005.
[17] R. C. Merkle, "Protocols for public key cryptosystems," *Security and Privacy, IEEE Symposium on*, vol. 0, p. 122, 1980.
[18] W. Zhang, M. Tran, S. Zhu, and G. Cao, "A random perturbation-based scheme for pairwise key establishment in sensor networks," in *MobiHoc '07.*
[19] M. Albrecht, C. Gentry, S. Halevi, and J. Katz, "Attacking cryptographic schemes based on "perturbation polynomials"," Cryptology ePrint Archive, Report 2009/098, 2009, http://eprint.iacr.org/.
[20] W. Zhang, H. Song, S. Zhu, and G. Cao, "Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks," in *ACM MOBIHOC*, May 2005.