ISSN:2320-0790

COMPUSOFT
An International Journal of Advanced Computer Technology

Secure Data Sharing with ABE in Wireless Sensor Networks

Dayananda RB[1] , Prof. Dr. G.Manoj Someswar[2] ,T.P. Suryachandra Rao[3]

[1]Associate Professor, Department of CSE, RRIT, Bangalore–90, Karnataka, India

[2]Principal & Professor, Department of CSE, Anwar-ul-uloom College of Engineering & Technology (Affiliated to JNTU, Hyderabad), Vikarabad – 501101, RR District, Telangana, India

[3]Principal, IMS PG Centre, Uppariguda (V), Ibrahimpatnam Mandal, R. R. District, Telangana, India

Abstract: This research paper addresses the issue of secure data sharing for distributed data storage in Wireless Sensor Networks (WSNs). In WSNs, storing data at local sensor nodes or at designated in-network nodes greatly saves the network-wide communication load and has a lot of benefits such as energy-efficiency. However, unattended wireless sensor nodes are very likely subject to strong attacks such as physical compromise. In this sense a storage node in WSNs can be viewed as an untrusted storage since the owner of the WSN may have concerns on data security in mission-critical applications if data are stored without proper protection. A secure data storage and retrieval scheme is required for distributed data storage in WSNs. When previous works focus on data confidentiality and integrity protection or communication security, the issue of fine-grained data access control in WSNs is seldom addressed. In this chapter we address this issue and provides a cryptographic-based access control mechanism with ABE. The main challenge in this work is to make the expensive ABE operations affordable to resource-constrained sensor nodes. We resolve this issue by dividing the lifetime of sensor nodes into phases and then distribute the underlying mathematical operations in ABE over these phases. To minimize the communication and computation load on sensor nodes in case of user revocation, we revise an existing ABE scheme and makes the user revocation complexity on sensor nodes constant. Formal security proof and experimental results shows that our proposed solution is provably secure and affordable to real sensor nodes. To the best of our knowledge, our work is de facto the first that provides a secure mechanism for distributed fine-grained data access control in WSNs.

Keywords: *Wireless sensor networks, Fine-grained Data Access Control, Collusion Resilience, Sensor Compromise Resistance, Backward Secrecy, Access Control Strategy, Master Key Encryption*

I. INTRODUCTION

In this research work, we consider a wireless sensor network composed of a network controller which is a trusted party, a large number of sensor nodes, and many users. Throughout this research work, we denote the network controller with the symbol T. Symbol *U* and *N* are used to represent the universe of the users and the sensor nodes respectively. Both users and sensor nodes have their unique IDs. Symbol *Ui* will be used to denote user *i*, and Nj to represent sensor node *j*. The trusted party *T* can be online or off-line. It comes online merely on necessity basis, e.g., in the case of intruders detected. Each sensor could be a high-end sensor node such as iMote2 which has greater processing capability and a larger memory than conventional sensor nodes. Sensor data could be stored locally or at some designated in-network locations. As is conventionally assumed, we consider each user *Ui* to have sufficient computational resources to efficiently support expensive cryptographic operations such as bilinear map. In addition, we assume there is a loose time synchronization among the sensor nodes, and the lifetime of the network is further divided into phases based on the time synchronization.

II. DESIGN & IMPLEMENTATION

Adversary Model

This research paper considers attackers whose main goal is to learn about the contents of sensor data that they are not authorized to. The adversaries could be either external intruders or unauthorized network users. Due to lack of physical protection, sensor nodes are usually vulnerable to strong attacks. In particular, we consider the adversary with both passive and active capabilities, which can (1) eavesdrop all the communication traffics in the WSN, and (2) compromise and control a small number of sensor nodes. In addition, (3) unauthorized users may collude to compromise the encrypted data.

Security Requirements

For the purpose of securing distributed data storage in WSNs, the main goal of this work is to protect contents of sensor data from being learned by attackers, including external intruders and unauthorized network users. With respect to data access control in WSNs, we recognize the following unique but not necessarily complete security requirements.

Fine-grained Data Access Control: In many application scenarios, especially mission-critical cases, disclosure of sensitive data should be well controlled such that different users many have access privileges over different types of data.[1] For this purpose, we need to define and enforce a flexible access policy for each individual user based on the user's role in the system. In particular, the access policy should be able to define a unique set of data that the user is authorized to access, and must be enforced via a cryptographic method since sensor nodes are vulnerable to strong attacks like physical compromise.

Collusion Resilience: As described by our adversary model, unauthorized users may cooperate for the purpose of learning about the contents of sensitive data. [2]This requires our data access control scheme to be resilient to collusion attacks in the sense that the collaboration of unauthorized users will not give them additional advantages over what they can directly obtain from executing attacks individually.

Sensor Compromise Resistance: Due to lack of compromise-resistant hardware, a small number of sensor nodes could be physically compromised by the adversary in hostile environments. Now that the adversary can always obtain the sensor data generated by a sensor node after it is compromised, we should at least secure sensor data such that, (1) compromising the sensor node does not disclose the sensor data generated before the sensor is compromised, and (2) compromising one sensor node does not give the adversary any assistance to obtain sensor data generated by other sensor nodes.[3]

Backward Secrecy: User management is an important functionality required by most application scenarios. In particular, the system should be able to handle user revocation in the case of user leaving request or malicious behavior detected. To support such a functionality, the data access control mechanism should guarantee that the revoked users are not able to access the sensor data generated after they are revoked.

Our Proposed Scheme

This research work presents data access control scheme for distributed data storage in WSNs. We first introduce our access control strategy. Next, we give an overview of our proposed scheme. Then, we present the detailed description of our basic scheme, which is followed by an advanced design.

Access Control Strategy

For the purpose of achieving fine-grained data access control in WSNs, we first explore some inherent natures of WSNs. In general, the deployments of most WSNs are aimed at collecting certain types of data for specific application(s). Therefore, we are able to specify individual sensors (and hence the data collected by them) through a set of predefined attributes. For example, in the battlefield, sensor nodes are usually deployed to collect military information in certain geographic location. [4]Each sensor node may be responsible for collecting specific types of data such as vibration, smoke, so on and so forth. Sensor nodes may also have their owners, i.e., persons or units who are in charge of them. In particular, some nodes may be jointly owned by different units. Hence, we may specify sensor nodes using these attributes such as {location = *village*, data type = *(vibration, smoke)*, owner = *(explosion experts, officers, scouts)}*. This further enables us to specify data access privileges of users based on these attributes. In the above example, we may designate the access structure of a user as "(location is village) AND (type is vibration)", which allows the users to obtain vibration data within the village area. We may also define more sophisticated access structures such as "(location is village) AND (type is vibration OR smoke) AND (at least owned by 2 of the following: explosion experts, officers, scouts)". In this case, the user can only access vibration and smoke data collected within the village area. In addition, the last condition implicitly requires the user to belong to at least two of the three designated groups. To enforce these access structures, we predefine a public key component for each of the attributes, and encrypt sensor data with public key components of the corresponding attributes such that only the users with "satisfiable" access structures[14] are able to decrypt.

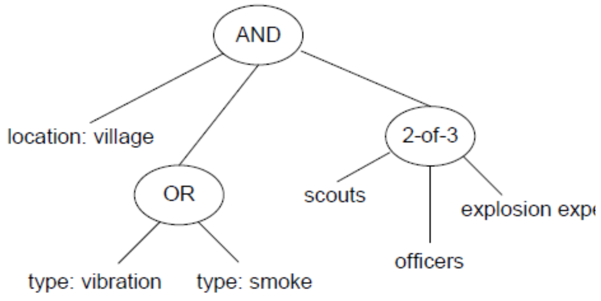

Figure 1: An example access structure in the battlefield scenario

Having discussed the intuitive idea of our data access control strategy, we further present it more formally as follows. In our proposed scheme, we associate each sensor node (and hence its collected data) with a set of attributes, for each of which we define a public key component. Each user is assigned an access structure, which is implemented via an access tree and embedded in the user's secret key. Every leaf node of the access tree is labeled with an attribute and the interior nodes are defined as threshold gates. This kind of definition of user access structure is able to represent very expressive logic expressions over attributes, and thus specify data access privileges of users in the fine-grained manner. Actually, we are able to represent any general (monotone or non-monotone) access structures if we define the NOT of each attribute as a separate attribute, which in turn will double the number of attributes in our system. Figure 1 illustrates the aforementioned access structure in the battlefield scenario.

Formally, in this work we will denote the universe of all the sensor attributes in a WSN by a symbol I. The set of attributes owned by each single sensor node is denoted by a symbol I_i, where i is the sensor node ID. We have $\mathcal{I} = \bigcup_{\forall i \in \mathcal{N}} \mathcal{I}_i$. Let $k = \max_{\forall i \in \mathcal{N}} |\mathcal{I}_i|$ k will be a system parameter used by our scheme. The access structure is generally denoted by *P*.

III. Scheme Overview

In our basic scheme, each sensor node is preloaded with a set of attributes as well as the public key *PK*. Each user is assigned an access structure and the corresponding secret key *SK*. The lifetime of the sensor network is divided into *phases*, each of which has the same time duration. Based on this, we further define each *n* consecutive phases as a *stage*, where $n < k$ and $k = \max_{\forall i \in \mathcal{N}} |\mathcal{I}_i|$ is a system parameter. Therefore, the lifetime of the sensor network can also be represented by a series of consecutive *stages*, numbering as 1, 2, …, *m*, where *m* is a system parameter. Sensor nodes encrypt sensed data using a symmetric-key algorithm, e.g., AES. Over each phase every sensor node updates its data encryption key once in the way that the data encryption keys during one stage form a one-way key chain. The key update algorithm could be any standard one-way hash function such as SHA-1. We call the first key on this key chain by the *master key*, denoted by *K*. The master key of each stage is always generated during its preceding stage, and encrypted under the preloaded attributes. Upon request for sensor data, the sensor node responds with the encrypted master key as well as the ciphertext of the sensor data.[5] If the user is an intended receiver, he is able to decrypt the master key and derive the data encryption keys for phases of his interest, and thus decrypt the sensor data. Based on the basic scheme, our advanced scheme goes one step further by providing the functionality of user revocation, which is demanded by most WSN application scenarios. In the advanced scheme, *T* is able to revoke any user via broadcasting a user revocation message to all the users and all the sensor nodes respectively. In particular, the user revocation message for the sensor nodes contains merely one group element of *GT*.

The Basic Scheme

The construction of our basic scheme based on KP-ABE and the one-way key chain is as follows:

System Initialization

On initialization, *T* executes the following steps:

a) Select two multiplicative cyclic groups G_1 and GT of prime order *p* as well as a bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Let *g* be the generator of G_1.

b) Choose a number t_i uniformly at random from Z_p for each attribute $i \in \mathcal{I}$, and *y* randomly from Z_p. Output the public key as follows:

$$PK = \langle G_1, g, Y = e(g,g)^y, T_1 = g^{t_1}, \cdots, T_{|\mathcal{I}|} = g^{t_{|\mathcal{I}|}} \rangle$$

The master secret key is $MK = (y, t_1, \cdots, t_{|\mathcal{I}|})$

c) Choose a secure one-way hash function, denoted as *h(-)*. Pre-load the following information to each sensor node N_i:

$$\mathcal{T} \rightarrow \mathcal{N}_i : \mathcal{I}_i, \ h(\cdot), \ PK$$

d) For each user *Uj*, *T* generates an access structure *P* and computes his secret key *SK* as follows. Starting from the root node r of *P* and in the top-down manner, construct a random polynomial q_x of degree d_x + 1 using Lagrange interpolation for each node x in P, where d_x is the threshold value of node *x*. For each non-root node *x* in *P*, set $q_x(0)$ = q_{pa}*rent(x) (index(x))*, where *parent(x)* is the parent of *x* and *index(x)* is the unique index number of *x* given by its parent. In particular, set $q_r(0) = y$. *SK* is output as follows:

$$SK = \langle \{D_i = g^{\frac{q_i(0)}{t_i}}\}_{i \in \mathcal{L}} \rangle$$

where *L* denotes the set of leaf nodes in *P*. Then, *Uj* is pre-loaded with the following information

$$\mathcal{T} \rightarrow \mathcal{U}_j : \mathcal{P}, \ SK, \ h(\cdot), \ PK$$

Master Key Encryption

During each stage $v \in [1, m], \mathcal{N}_i$ generates a new master key for stage *v* + 1 and encrypts it as follows:

a) Select a number *s* uniquely at random from Z_p.

b) On each phase of stage *v*, calculate one item $E_i = T_i^s$ for attribute $i \in \mathcal{I}_i$.

After $|\mathcal{I}_i|$ phases, $|\mathcal{I}_i| \leq k$, $\mathcal{N}_i$ has the complete se

c) Randomly select a number $K \in \mathcal{K}$ as the master key of the key chain, where *K* denotes the key space. Then, compute $E' = KY^s$. Finally, store the ciphertext as follows:

$$E^{v+1} = \langle v+1, \ \mathcal{I}_i, \ KY^s, \ \{E_i = T_i^s\}_{i \in \mathcal{I}_i} \rangle$$

where E^{v+1} represents the encrypted master key for the $(v+1)^{th}$ stage.

Data Storage

Ni encrypts and stores the sensor data generated in the current phase, say phase $t \in [1, n]$ of stage $v \in [1, m]$, as follows:

a) Calculate the data encryption key $K_t = h(K_{t-1})$. In particular, we set $K_0 = K$.

b) Encrypt the sensor data, denoted by *D*, with current data encryption key K_t. Then, store the item $\langle v, t, \{D\}_{K_t} \rangle$, where $\{D\}_{K_t}$ represents the encrypted
sensor data.

c) Erase K_{t-1} from the memory.

For each sensor node, all the data encryption keys used during one stage form a one-way key chain. The sensor node just keeps the latest data encryption key in its memory, while erasing all the previous ones.

Data Access

Assume user U_j is requesting for sensor data generated by sensor node *Ni* during phase *t* of stage *v*. N_i responds the data query request with the following message:

$$\mathcal{N}_i \rightarrow \mathcal{U}_j : \langle E^v, \ \{D\}_{K_t} \rangle$$

On receiving the response from N_i, U_j executes the following steps to obtain the sensor data:[6]

a) Decrypt the master key *K* of stage *v* from E^v. The decryption process starts from the leaf nodes and in the bottom-up manner. First, U_j computes the value F_i for each leaf node *i* in *P* as follows.

$$F_i = \begin{cases} e(D_i, E_i) = e(g, g)^{sq_i(0)}, & \text{if } i \in \mathcal{I}_i \ ; \\ \perp, & \text{otherwise.} \end{cases}$$

Then, it proceeds in the recursive way from the second last layer as follows: for node *x* which is a d_x-*of*-*n* gate, if more than *n* – d_x children returns ⊥ L, $F_x = \perp$. Otherwise,

$$F_x = \prod_{i \in S_x} F_i^{\delta_i(0)} = \prod_{i \in S_x} e(g,g)^{sq_i(0)\delta_i(0)} = e(g,g)$$

where S_x denotes the set of *x's* children and $\delta_i(0)$ is the Lagrange coefficient which can be calculated by the user himself. If *P* "accepts" $\mathcal{I}_i, \mathcal{U}_j$ will finally obtain $e(g,g)^{sq_r(0)} = e(g,g)^{sy}$ and thus decrypt the master key *K*. Otherwise, the decryption algorithm returns $\perp$.

b) If the decryption algorithm returns $\perp$, terminate. Otherwise, U_j calculates the data encryption K_t from *K* by $K_t = h^t(K)$., and finally decrypts the sensor data with K_t.

In this basic scheme, we assign each sensor node a set of attributes and each user an access structure. Sensor data are encrypted under the attributes such that only the users with "satisfiable" access structures are able to decrypt. As the access structure is very expressive, we are able to precisely control the access privilege of each user, and thus enjoy fine-grained data access control. The access policies in the basic scheme are actually enforced by using KP-ABE. To alleviate the computation overload, we divide the lifetime of sensor nodes into stages and phases. On each stage a master key is generated to serve as the "seed" for the data encryption keys of the underlying phases. For the purpose of access control, we just need to encrypt the master key of each stage under the attribute-based encryption algorithm. Sensor data are encrypted using symmetric-key encryption such as AES which is very efficient.[7] As master keys are generated at a relative low frequency, we are able to distribute the computation overload of attribute-based encryption into each phase and thus make the expensive operations affordable to the sensor nodes.

User Revocation

Another fundamental functionality of WSNs is user management. In particular, we stress that the network operator should be able to revoke the user's access privilege when necessary. In our basic scheme, we can use the following approaches to revoke users from the system: one approach is to define some time attributes and embed an expiration date to each user's access structure based on the time attributes.[8] Sensor nodes can then associate a time stamp to each ciphertext using the time attributes. If sensor nodes always associate the current time stamp to cipher-texts, users will be automatically revoked after their designated expiration dates. Another approach for user revocation is to define some "identity attributes", e.g., defining a binary attribute for each bit of user identity, and associate the corresponding identity attributes to each user's access structure. Sensor nodes can then associate any intended user list with each ciphertext using the "identity attributes". To revoke a user, sensor nodes can encrypt data using a selected set of "identity attributes" which exclude the revoked user's identity. The advantage of the two approaches is that they do not involve extra communication with users. However, the limitation of them is also obvious.[9] For the first approach, users can only be revoked at a pre-defined time. It does not support user revocation on the fly. The second approach is "stateful", i.e., every ciphertext (and hence the sensor nodes) needs to remember all the revoked users in the history. The ciphertext size would keep increasing as more and more random users are revoked, which ends up with a heavy computation and communication overhead on each sensor node after several rounds of user revocation. To resolve this issue, in this work we propose to update secret keys of all the users but the one(s) to be revoked. More specifically, we will update a common master key component which is embedded into every user's secret key as we will discuss in detail. The benefits of this key update method can be summarized as follows. First, this approach is "stateless" and sensor nodes do not need to "remember" any revoked user in the history. Second, the user revocation process does not introduce too much communication or computation overhead on each sensor node. Actually, each sensor node just needs to update one of its public key components which can be efficiently achieved by broadcasting the common public key component to all the sensor nodes. Consequently, the affect of user revocation on each sensor node is minimal. The main issue with the proposed solution, however, is that it needs every user to communicate with the authority via unicast to update his secret key. To resolve this issue, we revise the original KP-ABE construction so that we can update secret keys for all non-revoked users by broadcasting a common element to them, which can be efficiently realized by existing broadcast encryption techniques. We also prove that our revision to KP-ABE has the same security strength as the original construction in terms of semantic security of data.

IV. The Advanced Scheme

The basic idea of our advanced user revocation solution is to separate the master secret key *y* from the user access structure in the user secret key *SK*. Update of user secret keys can thus be realized by updating the embedded secret *y* which is common to every user's secret key. [10]As a result, we can update user secret keys via broadcasting the incremental of *y* while excluding the leaving user from the recipient list. Based on this general idea, we present our advanced scheme as follows. For brevity, we just present the parts that need to be changed as compared to our basic scheme.[11]

System Initialization

T executes the following steps.

a) The same as step a) of 1) in the basic scheme.

b) In addition to the elements generated by step b) of 1) in the basic scheme, *T* selects a number *β* uniquely at random from Z_p. The public key *PK* and the master secret key *MK* are then output as follows.

$$PK = \langle G_1,\ g,\ Y,\ \{T_i = g^{t_i}\}_{i\in\mathcal{I}},\ g^{\beta}\rangle$$

$$MK = \langle y, t_1, \cdots, t_{|\mathcal{I}|}, \beta\rangle$$

c) The same as step c) of 1) in the basic scheme.

d) Sensor node *Ni* is pre-loaded with the following

$$\mathcal{T} \rightarrow \mathcal{N}_i:\ \mathcal{I}_i,\ h(\cdot),\ PK$$

e) The process of key generation is similar to step d) of 1) in the basic scheme. *T* outputs the user secret key *SK* as follows.

$$SK = \langle g^{\frac{y-\theta}{\beta}}, \{D_i = g^{\frac{q_i(0)}{t_i}}\}_{i\in\mathcal{L}}\rangle$$

Compared to the basic scheme, this algorithm introduces a new element *gβ* into *SK,* where $\theta = q_r(0)$ is randomly selected from Z_p, and q_r denotes the polynomial for the root node r in *P*. *Uj* is then preloaded with *hP, SK, h(-), PK).*

Master Key Encryption

Similar to 2) in the basic scheme. The advanced scheme introduces a new element $g^{\beta s}$ into the ciphertext as follows:

$$E^{v+1} = \langle v+1, \mathcal{I}_i, KY^s, \{E_i = T_i^s\}_{i\in\mathcal{I}_i},\ g^{\beta s}\rangle$$

Data Storage

The same as 3) in the basic scheme.

Data Access

This part is the same as 4) in the basic scheme except for step a).

a) The decryption process is similar to that in the basic scheme.[12] When the data attributes satisfy the user's access structure *P*, the user obtains $e(g,g)^{\theta s}$. Then, he decrypts the message as follows.

$$M = \frac{Me(g,g)^{ys}}{e(g^{\frac{y-\theta}{\beta}}, g^{\beta s})e(g,g)^{\theta s}}$$

In this advanced scheme, *T* is able to update the master secret key *y* embedded in the user secret key *SK* by broadcasting $g^{\frac{\Delta y}{\beta}}$ to the users, where *Δy* is the incremental of *y*. With the above enhancement, we can present our user revocation scheme as follows.

User Revocation

To revoke a user *Uj*, *T* needs to update the master secret key *y* for the sensor nodes as well as the remaining users. The process can be illustrated as follows.

$$\mathcal{T}:\ y' \leftarrow \mathbb{Z}_p,\ \Delta y \leftarrow y' - y,\ Y' \leftarrow e(g,g)^{y'},\ \ g^{\frac{\Delta y}{\beta}}$$

$$\mathcal{T} \rightarrow \mathcal{N}:\ Y'$$

$$\mathcal{T} \rightarrow \mathcal{U}\backslash\mathcal{U}_j:\ g^{\frac{\Delta y}{\beta}}$$

First, *T* chooses a random number $y' \in \mathbb{Z}_p$ as the new value of the master secret key *y*. The incremental is set as $\Delta y = y' - y.$. Then, it calculates the new public key $Y' = e(g,g)^{y'}$ and the group element $g^{\frac{\Delta y}{\beta}}$. Finally, *T* broadcasts *Y'* to all the sensor nodes and $g^{\frac{\Delta y}{\beta}}$ to all the users excluding the one to be revoked. Upon receiving the master secret key update message, each sensor node simply replaces the public key *Y* with *Y'*. The master key for the next stage will be encrypted under the new public key. Each user updates his secret key as follows:

$$g^{\frac{y-\theta}{\beta}} g^{\frac{\Delta}{\beta}} = g^{\frac{y'-\theta}{\beta}}$$

The master secret key y is thus updated as y'. In this user revocation scheme, one challenging issue is to selectively broadcast $g^{\frac{\Delta y}{\beta}}$ such that all but the leaving users are able to receive it. Fortunately, there are plenty of off-the-shelf selectively broadcast schemes available for different application scenarios. is able to broadcast any $n — r$ out of n users with ciphertext size of $O(r)$ and private key size of $O(log^2n)$, which is further reduced to $O(logn)$ by. This scheme is suitable for application scenarios where the number of revoked users each time is small. Boneh et al. proposed a scheme which is able to broadcast to arbitrary subset of users with constant ciphertext size (only two group elements). This scheme is extremely suitable for bandwidth-critical applications. One drawback of this basic scheme of is that the public key size is of $O(n)$. To balance the size between the public key and the ciphertext, a revised scheme is presented in which both the ciphertext and the public key are of size $O(\sqrt{n})$. Cheung et al. proposed a collusion-resistant broadcast encryption scheme based on flat table scheme and attribute-based encrytpion. Both the ciphertext and the user secret key are of size $O(logn)$. Yu et al. further improved by supporting receiver anonymity. and are suitable for scenarios in which the system wants to revoke users of some common attributes, or the number of revoked users each time is small. In our proposed scheme, we do not designate any particular selective broadcast scheme for user secret key update.[13] The system designer can pick an appropriate broadcast scheme from the above candidates according to the requirement of the actual system.

V. RESULTS & DISCUSSIONS

Change of Sensor Attributes

Conventionally the set of attributes of each sensor node does not change throughout the node's lifetime, or we can make this assumption as it is enough for many application scenarios. Nevertheless, there are still some cases in which the attributes of sensor nodes would change. For example, in some dynamic environments such as battlefields, the location of a portion of sensor nodes might be adjusted frequently. In this case, it is desirable to change the location attributes for the involved sensor nodes while not affecting the others. To achieve this goal, we just need to load the involved sensor nodes with the new attribute lists as well as the corresponding public key components. This can be easily realized in our proposed scheme as long as the involved sensor nodes are convinced of the authenticity of the update, which can be realized without any difficulty by attaching the network controller's signature to the update.

Support for Concealed Data Aggregation

In-network aggregation of data has been put forward as an important paradigm for wireless sensor networks which enables in-network consolidation of redundant data and thus saves energy. In practical settings, it is often desired to provide in-network data aggregation while guaranteeing data confidentiality for privacy concerns. The concept of Concealed Data Aggregation (CDA) was proposed to address this issue. With CDA, the intermediate nodes are able to aggregate data by performing the aggregation operations on incoming ciphertexts without knowing the data encryption keys nor the plaintext. To realize CDA, sensing nodes should encrypt data using certain encryption transformation, a.k.a. *privacy homomorphism (PH).* A survey on existing PH schemes, including symmetric and asymmetric ones, can be found in. We stress that our proposed scheme can seamlessly integrate existing symmetric PH schemes and thus realize CDA. To justify this, we take as an example and show how that PH scheme is integrated with our proposed scheme. At a high level the PH scheme in has the property as follows. Given two messages $m1$ and m_2, and their respective encryption keys k_1 and k_2, if $c1 = Enc(m1, k1)$ and $c_2 = Enc(m2, k_2)$, then $c1 + c_2 = Enc(m1 + m_2, k1 + k_2)$ and $Dec(c1 + c_2, k1 + k_2) = m1 + m_2$. This property still holds for the case of more than two messages and can be used to compute statistical values, e.g., mean, variance and standard deviation, of sensed data. Intuitively, we can integrate this PH scheme into our proposed scheme in the following way: First, let each sensing node encrypt the sensed data with its data encryption key k_i and encrypt k_i (actually its seed) under its attributes. This process is basically the same as that of our proposed scheme. Then, upon data query every sensing node sends both the ciphertext of the sensed data and that of k_i to its upstream aggregating node. The intermediate aggregating nodes, after having collected all the downstream data, do the aggregation operations on the ciphertexts of data while keeping ciphertexts of the data encryption keys intact. Subsequently, they transmit the aggregated ciphertext of data along with the ciphertexts of data encryption keys to their respective upstream aggregating nodes. The above process is recursively executed until it reaches the user. The user, on receiving the ciphertexts, first recovers the data encryption keys if his access structure satisfies with the sets of attributes of all the sensing nodes. Then he does the same aggregation operations over the recovered data encryption keys as all the aggregating nodes did to compose a "aggregated" data encryption key of the final data ciphertext and decrypt the aggregated value of data.

The drawback of this intuitive solution is that the ciphertexts of the data encryption keys could be a heavy communication overhead. This is because the size of such a ciphertext grows linear with the number of attributes of the sensing node and each intermediate aggregating node should forward these ciphertexts of all its downstream sensing nodes. To alleviate this overhead, we have enforced our access control strategy only on few designated upstream aggregating nodes. In this way the downstream sensing nodes just need to encrypt sensed data with their data encryption keys. These designated upstream aggregating nodes fulfill our access control strategy by encrypting the "aggregated" data encryption keys under certain set of attributes. One issue underlying this method is that the aggregating nodes should distribute data encryption keys to all its downstream sensing nodes, which can be resolved using existing key distribution methods.

Scheme Evaluation

This research work evaluates our proposed scheme in terms of security and performance aspects.

Security Analysis

We evaluate the security of our work by analyzing the fulfillment of the security requirements:

Fine-grained Data Access Control: To provide fine-grained data access control, the proposed scheme should provide a strategy that is able to define and enforce complex access policies for sensor data of various types or security levels. In FDAC, the access structure embedded in each user's secret key is able to represent complicated predicates such as disjunctive normal form (DNF), conjunctive normal form (CNF), and threshold gates. The combination of these predicates are able to represent sophisticated access structures. In fact, our scheme is able to support non-monotonic (general) access structures if we define the *NOT* of each attribute as a separate attribute, which in turn will double the number of attributes in our system. To enforce our access control strategy, we encrypt the master key of the key chain in each stage under a set of attributes. Without the master key, the adversary is not able to derive the data encryption keys due to the one-wayness of the key chain, which can be guaranteed by choosing a secure one-way hash function such as SHA-1. In our basic scheme, the master key is actually encrypted under the standard key-policy attribute-based encryption (KP-ABE) scheme which is provably secure. Our advanced scheme, to achieve efficient user revocation, makes some enhancement to the standard KP-ABE when encrypting the master key. The enhanced KP-ABE is provably secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. (A formal security proof is available in our thesis). This turns out that the adversary is not able to decrypt the master key unless he owns the intended access structure. Therefore, our proposed scheme is able to control the disclosure of sensor data so that only authorized users are able to access.

Collusion Resilience: To compromise sensor data, the main task of the colluding users is to decrypt the master key of the target data if the one-wayness of the underlying one-way has function, e.g., SHA-1, is guaranteed. Since the master key is encrypted under our enhanced scheme, we have to prove that it is collusion-resistant. Intuitively, we can sketch the collusion-resistance of our enhanced scheme as follows. Recall that the master key is encrypted in the form of $Ke(g,g)^{ys}$. The user has to cancel $e(g,g)^{ys}$ to recover K. To compose $e(g,g)^{ys}$, the only way is to execute the following:

$$e(g^{\frac{y-r}{\beta}}, g^{\beta s}) = e(g,g)^{ys}/e(g,g)^{rs}.$$

To extract $e(g, g)^{ys}$, the user should compute $e(g, g)^{rs}$. Actually, for each user, r is randomly and independently selected from Z_p. The secret key from one unauthorized user does not give the other user any help in terms of computing $e(g, g)^{rs}$. Actually, in our security proof the security definition implies collusion resistance. As our scheme is provably secure under this security definition, collusion resistance is also guaranteed.

Sensor Compromise Resistance: To meet this security requirement, we should achieve two securitygoals:
(1) compromising the sensor node does not disclose the sensor data generated before the sensor is compromised,and
(2) compromising one sensor node does not give the adversary any advantage to obtain data generated by other sensor nodes. We can show the fulfillment of our scheme with respect to these two security goals as follows: (1) In our scheme, each sensor node just keeps the current data encryption key in the memory, while erasing all the previously used keys. Because of the one-wayness of the key chain, the compromiser is not able to derive the previously keys from the current key. (2) is easy to prove since each sensor node encrypts data independently.

Backward Secrecy: As is described in the previous section, our advanced scheme is able to update the master key *y* for legitimate users while excluding those to be revoked. Since the new sensor data will be encrypted under the latest master key, the revoked users are not able to decrypt. One problem in our scheme is, the user revocation instruction will not take effort until a new stage starts. Such a delay occurs because it would take one stage for each sensor node to encrypt the master key under the attributes. This delay may differ for different systems. For example, if a system defines 30 phases as a stage and each phase lasts 1 second, the delay will be at the most half a minute. Generally, if a system has a stage with less phases and each phase takes less time, e.g., each sensor node is assigned a smaller number of attributes or has a more powerful computational capability, the delay can be shorter. In practical applications, we leave this delay as a system parameter, and the system designer can adjust this parameter by changing the number of attributes assigned to each sensor node or using a different type of sensor nodes.

In addition to the security goals listed above, there are also some other security requirements such as data integrity and authenticity, which are desired by conventional WSN applications. In fact, security requirements such as message integrity can be easily supported in our scheme with minor modifications using existing off-the-shelf techniques. A challenging orthogonal issue would be data authenticity which requires sensing nodes to provide proofs of data authenticity to users. Some current work such as has provided salient solutions to this problem. As the main focuses of this work is fine-grained data access control, we do not explicitly address all those security problems.

VI. Performance Evaluation

This research paper evaluates the performance of our proposed scheme in terms of computation and communication overheads. In our scheme, sensor data are generated and encrypted by sensor nodes, and retrieved and decrypted by users. As sensor nodes are usually resource constrained, they may not be able to execute expensive cryptographic primitives efficiently and thus become the bottleneck of the scheme. For this reason, our evaluation focuses on the performance of sensor nodes. In the following section, we first discuss the numeric results in terms of computation and communication overheads for sensor nodes. Then, we present our implementation results on real sensors.

VII. Numeric Result

In our proposed scheme, each sensor node is responsible for the following operations in each stage:
(1) generate the master key and encrypt it using our enhanced KP-ABE, (2) derive the data encryption keys based on the master key, and (3) encrypt sensor data using the data encryption keys. These operations are further distributed to each phase. Specifically, if we choose elliptic curves as the underlying bilinear group, in each phase the sensor node needs to execute at the most one scalar multiplication on elliptic curves, one one-way hash, and one symmetric key data encryption. Table 2.8 lists all these operations.

Table 1: Computation load on each sensor node

	Scalar Mul	Hash	Data Encryption
Each Stage	$\lvert\mathcal{I}_i\rvert + 1$	n	n
Each Phase	1 or 0	1	1

On each data retrieval request, the sensor node responds with $\langle E^v, \{D\}_{K_t}\rangle$ for sensor data of phase *t* in stage *v*, where E^v contains $\lvert\mathcal{I}_i\rvert + 1$ group elements on G_1 and one on G_T, and $\{D\}_{Kt}$ is the data payload. On user revocation, *T* only needs to broadcast one group element of G_T to all the sensor nodes. The communication overload for each sensor node is shown in Table 2.9.

Table 2: Communication load

Data Retrieval	User Revocation
$(\lvert\mathcal{I}_i\rvert + 1)\ \mathbb{G}_1 +\ 1\mathbb{G}_T$ + data payload	$1\mathbb{G}_T$

VIII. Implementation

In our implementation, we choose Tmote Sky and iMote2 as the target platforms. We use SHA-1 as the one-way hash function and AES (supported by CC2420 Radio module of the motes) as the data encryption algorithm. Our implementation shows that it takes about 0.06ms for SHA-1 to execute one hash operation and 0.4ms for AES to encrypt 64 bytes data. Our implementation also shows that one scalar multiplication takes several seconds in the worst case. The scalar multiplication operation is thus the bottleneck of the sensor performance. To optimize this operation, one key issue is to find appropriate parameters for the elliptic curve.

In past years, many works have efficiently implemented Elliptic Curve Cryptography (ECC) on various sensor platforms. In these works, elliptic curves are ususally chosen according to standards such as NIST and SECG, which enable most of the optimization methods. Although these elliptic curves serve perfectly for security schemes such as ECDH, ECDSA, et al, they are not pairing-friendly, i.e., they cannot be used as bilinear groups. In FDAC, however, the elliptic curve is required to be pairing-friendly. Most pairing-friendly elliptic curves studied by current work fall into two categories, namely Supersingular (SS) curves and MNT curves. In the case of SS curves, the two elliptic groups G_1 and G_2 (cf. Section 2.2) could be the same. For MNT curves, G_1 and G_2 are different. To choose an appropriate elliptic curve, several factors should be taken into account as follows. Let l be the group size of the elliptic curve and *k* be its embedding degree. To achieve a comparable security strength of 1024-bit RSA, we should have *lk* to be larger than 1024, or at least close to 1024. Given the security level, a higher *k* results in a shorter group size. Therefore, choosing a high embedding degree for the elliptic curve in our scheme may result in not only a short ciphertext, but also an efficient scalar multiplications on each sensor. However, the embedding degree *k* of the elliptic curve cannot be arbitrarily large. Choosing an appropriate embedding degree for the elliptic curve is actually another research area. According to the benchmark of Pairing-Based Crypto (PBC) library, elliptic curves with l = 512 and *k* = 2 results in the fastest bilinear pairing as compared to those with *k* > 2 for SS curves. The case is on the opposite for MNT curves. According to our testing of the PBC library on Linux platform with an Intel Pentium D 3.0GHz CPU, SS curves with l = 512 and *k* = 2 (type *a* curves in PBC) take about 6ms to execute a pairing, while MNT curves with l = 159 and *k* = 6 (type *d* curves in PBC) take about 14ms (Actually, on the user side of our solution decryption time is linear to the number of pairings). Although both results are acceptable to users, MNT curves imply a much shorter ciphertext as well as key size to sensor nodes. More importantly, scalar multiplication over 512-bit curves may not be supported by low-end sensor nodes such as Tmote Sky because it consumes too much RAM. For these reasons, we believe MNT curves with high embedding degrees are suitable for our proposed scheme.

In our implementation, the elliptic curve is a MNT curve over F_q with embedding degree of 6, where *q* is a 159-bit prime number. The curve has the form $y^2 = x^3 + ax + b$. Our implementation is based on the TinyECC library with curve specific optimization disabled since the group size *q* is not a Mersenne prime. Our result shows that iMote2 consumes about 35ms to execute a scalar multiplication when working at 416MHz, 69ms at 208MHz, and 139ms at 104MHz. Tmote Sky consumes 4.1s. For the 512-bit SS curve, iMote2 consumes 170ms at 416MHz, 341ms at 208MHz, and 682ms at 104MHz. Tmote Sky does not have enough RAM to support 512-bit SS curve. Table 2.3 to Table 2.4 summarize the above implementation results.

Table 2.3: One-phase computation load on iMote2 (MNT curves)

SHA-1	AES	One Scalar Multiplication		
		104MHz	208MHz	416MHz
0.06ms	0.4ms	139ms	69ms	35ms

Table 2.4: One-phase computation load on iMote2 (SS curves)

SHA-1	AES	One Scalar Multiplication		
		104MHz	208MHz	416MHz
0.06ms	0.4ms	682ms	341ms	170ms

Table 2.5: One-phase computation load on Tmote Sk

SHA-1	AES	One Scalar Multiplication	
		MNT Curves	SS Curves
0.07ms	0.4ms	4.1s	N/A

We can estimate energy consumption of one phase using the equation $W = U \times I \times t$, where U is the voltage, I is the current draw, and t is the execution time for one phase. According to the date sheet of each platform the current draw for TMote Sky is *1.8mA* and the voltage can be chosen as 3v. The iMote2 data sheet just gives the current draw for running at 104MHz with radio on, which is *66mA*. To be conservative, we use this value in our computation. The voltage of iMote2 can be chosen as 0.95v. Based on the execution time we measured, the energy consumption on the iMote2 platform (running at 104MHz) for one phase is 8.74mJ in case of MNT curves and 42.79mJ in case of SS curves. The energy consumption on the TMote Sky platform for one phase is 24.68mJ (for MNT curves only).

IX. Further Enhancement

In the above user revocation scheme, $g^{\frac{\Delta y}{\beta}}$ is an update message common to all non-revoked users, which opens the door for a non-revoked user to collude with revoked users and help them decrypt the data. Boldyreva et al. proposed a user revocation scheme for IBE and KP-ABE in which user collusion attacks are well addressed. The proposed scheme is built on top of the construction of Fuzzy IBE and the binary tree data structure. More specifically, it introduces a time attribute and use it in the encryption of each message. The root node of each user's access tree is an AND gate with one child being the time attribute and the other being the root node of ordinary access structure. When a user is to be revoked, the system administrator generates key updates on the time attribute using the binary tree, each leaf node of which is associated to one user. Since new messages will be encrypted with the updated time attribute, users didn't receive the key updates will not be able to decrypt. In this scheme, the complexity of encryption and decryption is comparable to that of current KP-ABE. The complexity of user revocation in terms of message size and computation overhead is $O(rlog(\frac{n}{r}))$ when $1 < r \le n/2$, or $O(n-r)$ when $n/2 < r < n$, where r is the total number of revoked users and *n* is the total number of users. It should be noted that, we can also use this revocable KP-ABE in our scheme for achieving fine-grained data access control. One significant advantage of using the revocable KP-ABE is its enhanced security against user collusion. However, the complexity of user revocation is linear to the number of revoked users which could raise concerns in large scale systems when that number is approaching *n/2.* In our proposed user revocation solution, the system designer is free to choose a broadcast scheme. For example, he/she can use which has the constant ciphertext size if communication overhead is of the most importance. However, security level is reduced in this solution. We treat the above issue as a trade-off between efficiency and security and leave the choice to the system deployer.

References

[1] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST),* pages 29-42, 2003.

[2] S. Kamara and K. Lauter. Cryptographic cloud storage. In *Proceedings of the International Conference on Financial Cryptograpy and Data Security (FC),* pages 136-149, 2010.

[3] J. Li, Q. Huang, X. Chen, S.S.M. Chow, D.S. Wong, and D. Xie. Multi-authority ciphertext-policy attribute-based encryption with accountability. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS),* pages 386-390, 2011.

[4] Q. Liu, C.C. Tan, J. Wu, and G. Wang. Reliable re-encryption in unreliable clouds. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM),* 2011.

[5] Q. Liu, C.C. Tan, J. Wu, and G. Wang. Efficient information retrieval for ranked queries in cost-effective cloud environments. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM),* 2012.

[6] S. Miiller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In *Proceedings of Annual International Conference on Information Security and Cryptology (ICISC),* pages 20-36, 2009.

[7] B. Patel and J. Crowcroft. Ticket based service access for the mobile user. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom),* pages 223-233, 1997.

[8] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. *Journal of Computer Security,* 18(5):799-837, 2010.

[9] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proceedings of International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT),* pages 557-557, 2005.

[10] B. Stone and A. Vance. Companies slowly join cloudcomputing. *New York Times,* 18:2010, 2010.

[11] S. Subashini and V. Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications,* 34(1): 1-11, 2011.

[12] G. Wang, Q. Liu, and J. Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS),* pages 735-737, 2010.

[13] G. Wang, Q. Liu, and J. Wu. Achieving fine-grained access control for secure data sharing on cloud servers. *Concurrency and Computation: Practice and Experience,* 23(12):1443-1464, 2011.