# Distributed User Access Control in Sensor Networks*

Haodong Wang and Qun Li

Department of Computer Science
College of William and Mary
Williamsburg, VA 23187-8795, USA
{wanghd, liqun}@cs.wm.edu

**Abstract.** User access control in sensor networks defines a process of granting user the access right to the information and resources. It is essential for the future real sensor network deployment in which sensors may provide users with different services in terms of data and resource access. A centralized access control mechanism requires base station to be involved whenever a user requests to get authenticated and access the information stored in the sensor node, which is inefficient, not scalable, and is exposed to many potential attacks along the long communication path. In this paper, we propose a distributed user access control under a realistic adversary model in which sensors can be compromised and user may collude. We split the access control into local authentication conducted by the sensors physically close to the user, and a light remote authentication based on the endorsement of the local sensors. Elliptic Curve Cryptography (ECC), a public key cryptography scheme, is used for local authentication. We implement the access control protocols on a testbed of TelosB motes. Our analysis and experimental results show that our scheme is feasible for real access control requirement.

## 1 Introduction

Access control defines a process of identifying user and granting user the access right to information or resources. Sensor network is a computing platform for users to collect data, transmit data, and process data. The access control pertaining to sensor network predominantly aims to protect the network usage and collected data. Unauthorized user should not be allowed to use the network since network bandwidth is very limited and, more importantly, the battery power of each node may be depleted after malicious users aggressively effuse messages to the network. The data collected or processed, many times, is classified so that data of different classifications requires security clearance for authorized access. For example, a high rank officer may need to know more information about the

* This work was partially supported by the U.S. National Science Foundation under grant CCF-0514985.

P. Gibbons et al. (Eds.): DCOSS 2006, LNCS 4026, pp. 305–320, 2006.

field deployment than a soldier. In another scenario, information may be sensitive compartmented so that users have to be denied of access to the data that is beyond his access right. An example would be a user is authorized to access the data from the sensors in his office, but not other people's offices.

To achieve access control, it is essential for sensor nodes to authenticate the identities of the requesters. This paper aims to explore an efficient and secure authentication scheme for the sensor nodes. A natural way for the authentication check is to use a centralized mechanism. After receiving a request, the sensor node sends the user information to the base station. Then the base station decides whether the access is granted or not and replies the result to the sensor node. This solution may yield a good security result because of the fact that the base station is considered secure, and the communication channels between sensors and the base station are assumed secure. However, this scheme suffers two major problems. First, the centralized authentication requires at least one round-trip communication between the sensor and the base station. If a number of users are accessing the network at the same time, the authentication traffic may easily cause network congestion. Second, this authentication pattern is vulnerable to adversary's DoS attacks. The sensor nodes have no knowledge about user access right until they get replies from the base station. The adversary can easily launch DoS attacks by forging a large number of user access requests, which will in-turn trigger the same amount of authentication traffic. The consequence will severely saturate the network and quickly deplete the sensor node power.

This paper gives a thorough exploration for sensor network data access control problem in a general setting. We consider a data access scenario that a user can access in-network stored data at any location from anywhere in the network, which includes local data access from user's nearby sensors and remote data access. Moreover, we consider access control problem in a much harsher environment in which the users may collude and sensors may be compromised. Compromised sensors can get the information from the user authentication process and may disclose this information to an adversary, which may potentially help the adversary to gain more access privileges. Colluding users may analyze their information and design a scheme to counteract the access control system. Besides, we also addresses node duplication attack and DoS attack by inundating authentication messages to the network.

It is our belief that our more general data access model and realistic adversary threat model define a very realistic problem for future sensor deployment. Our work has following four contributions. First, we propose a practical and scalable certificate-based local authentication based on ECC. Public key cryptography eliminates the complicated key management and pre-distribution required by symmetric key schemes, and provides a very clean interface between the user and sensors. The advantage of certificate-based authentication is that sensors do not need the storage for user's public keys or a third party for public key verification. User public keys can be constructed from user certificates and published system information. Second, we propose a novel group endorsement scheme to

authenticate a user locally by a group of sensors and transfer the endorsement to the remote sensor. This scheme is resilient to limited number of compromised sensors and the DoS attack launched in the form of remote authentication. Third, our scheme eliminates the possibility of user collusion attack. The polynomial based secret sharing scheme proposed in [18] suffers user collusion attack. The collusion by a number of users can easily reconstruct the secret polynomial and reveal the system secrecy. Our certificate-based authentication is resilient to any user collusion attack. Fourth, we show our scheme is feasible in real sensor network deployment. We have implemented both local authentication and remote authentication on TelosB motes, which are based on our implementation of 160-bit ECC security primitives. Since the TelosB hardware multiplier is disabled in TinyOS, the computation is longer that it should be. It takes 3.1s to generate a public key and 10.8s to conduct local authentication.

## 2 Related Work

We believe that, with fast expanding sensor network technologies, more services will be available to allow direct interactions between users and sensor nodes. Obviously, the new communication paradigm poses more security challenges for small and power constrained sensor nodes. Different from the security problem in user access control we address in this paper, most related researches focus on secure and resilient communication links and resource management inside the networks.

Perrig *et al.* [12] construct $\mu$Tesla and introduce the asymmetric mechanism through a delayed symmetric keys disclosure: the base station broadcasts an encrypted message first, and then releases the secret key in scheduled time frame. Although KDC-based schemes suffer the scalability problem, broadcasting is still the basic, efficient to distribute or revoke secret keys in sensor networks.

Eschenauer and Gligor propose a random graph based key pre-distribution scheme [7]. The scheme assigns each sensor a random subset of keys from a large key pool, and allows any two nodes to find one common key and use that key as their shared symmetric key. Based on their contribution, a number of researches [3, 5] have delivered to strengthen the security and improve the efficiency. Since each sensor node only needs to store a small number of keys, the random graph based schemes have the advantage of scalability. However, in a sparse network or non-uniform distributed network, the key establishment could be difficult because a number of sensor pairs may not successfully finish pairwise key establishment.

Besides the above two types of security schemes, a number of research teams focus on the group key and authentication problems [17, 15, 1, 14, 6, 2]. Ye *et al.* [17] design a Statistical En-Route Filtering (SEF) mechanism to detect and drop false reports. The idea is to use probabilistic key sharing to authenticate the legitimate messages on the routing path. However, SEF cannot be used to authenticate the message sender because the remote sensor does not have enough knowledge (as the sink) to verify the message source.

Zhang *et al.* [18] propose several schemes to restrict and revoke the access privilege of a mobile sink. Their approaches are based on Blundo's scheme to establish secret key between the mobile sink and sensor nodes, and then use Merkle tree technique to reduce the overhead. The limitation of the scheme is that the mobile sink's moving track has to be predetermined by the base station. Compared with our scheme, we address a more general user/sensor communication problem. The mobile sink can be regarded as one type of special users in our scheme.

## 3 System Model

We consider a large scale wireless sensor network deployed in a variety of environments, e.g., at a hostile battlefield, in an office building, or in a national park. Data access to the stored data on each node is protected according to the attributes of the data, e.g., data type (temperature, light, noise, etc.), data location, data collection time, and so on. For a certain data, only authorized user can access the data from the storing node. Since the data is distributed in the entire network instead of in a central position, data protection by relying on a powerful sink node with all data access authorization information and computational power is not possible. Instead, data access authorization should be done in a distributed fashion accordingly. After the data access has been authorized, data access is granted to the user and data is transfered to the user.

A user equipped with a powerful computing device, such as a PDA, interacts with the sensor network for data query and retrieval and maybe network control such as network reconfiguration or sensing mode change. The PDA is the interface for the user to talk to the sensor network. The computing device is more powerful than the sensor nodes, so it is capable of more computationally intensive tasks. User can query data at any location of the network through sensor node relay. The data access capability, however, must be granted by a central authorization center before data access. A data access list is associated with the user about the types, locations, and the durations of the authorized data access. This information is encrypted in a way that the user is unable to forge and can be authenticated by the sensor holding the requested data.

The sensor network is managed by a Key Distribution Center (KDC), which is responsible for generating all security primitives (i.e., random numbers, one-way hash function, message authentication code (MAC), access list) and revoking users' access privilege if necessary. KDC distributes secret keys through the base stations. To access the sensor network, users need to apply for the access permission from KDC. KDC maintains a user access list pool and associated user identifications. The access list defines the user's access privilege. A typical access list is composed of *uid* and *user access privilege mask*. *uid* is a unique number to identify the user. *user access privilege mask* is a number of binary bits; each bit represents a specific information or service. An access list example is shown in Fig 1. The information stored at the sensor nodes is divided into multiple access privilege levels. The user with a lower access privilege is not allowed to get the

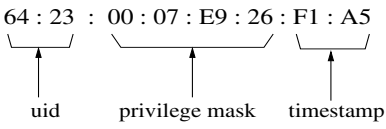



**Fig. 1.** An example of user access list. The access list is composed of three parts: *uid*, *access privilege mask*, and *timestamp*. *uid* is a unique number assigned to each user. *access privilege mask* is to define the user's access privilege to the system information. *timestamp* specifies the access list is only valid in a certain time frame.

information that requires the higher privilege. We assume the users can securely acquire their access lists from KDC through out-of-band secure communication channels. Once a user passes the authentication check, the sensor nodes provide their local information to the user. If the required information is not available locally, for the reason we will discuss later, a group of sensor nodes have to collaborate and request the information from the remote sensor which holds the information.

An adversary is assumed to use all possible means to access the data that is not authorized to him. He can eavesdrop message transmission to extract transmitted information or carry out message replay. Message eavesdrop and replay are easy to handle, as discussed by many papers, by using regular message encryption and including message sequence or time information. More hazard is created when nodes are compromised by the adversary who is able to garner all the information stored in the sensors. It is even worse that the adversary may inject his own program to the compromised sensors, which, under the control of the adversary, pretend to be trustworthy gaining as much information as possible. A user may also collude with the adversary for mutual benefit by attacking the access control system. The base station and the central authorization center cannot be compromised, however.

We mainly consider the following two potential attacks. First, Compromised sensors may capture much information and give to an unauthorized user so that that user may access data by impersonating another user. Second, user collusion may help users to subvert the system and gain more access right than that of anyone among the colluding users. We assume that at most $t$ sensors can be compromised. The assumption is reasonable because compromising sensors takes time and effort. On the other hand, we assume unbounded number of users can collude since it is not hard for mischievous users to share information and orchestrate an aggregated analysis to the collected information. The fact that a compromised sensor is hard to identify prevents a user from trusting any of the sensors. A user may have to disclose information for authentication, but the revealed information has to be specific to the sensor in contact and should not be used for authentication at another sensor.

We do not explicitly address the introduction of duplicated compromised sensors. However, since the duplicated compromised sensors do not introduce more information to the adversary, our carefully designed protocols do not enable the adversary to access the data from an uncompromised sensor.

## 4 Proposed Access Control Schemes

The user may request data stored locally or in a distant sensor. We first define following two types of sensor nodes. The sensor nodes which are directly within the contact range of the user are called *local* sensor nodes. The sensor nodes which cannot establish direct communication link with the user but hold the requested information are called *remote* sensor nodes.

In this section, we first propose a public-key cryptography based local access control scheme. Then we develop a remote access control approach (we assume that the ID of the remote sensor for data access is known by some scheme that is beyond the scope of this paper, e.g., resource discovery or geographic or location-based routing). Finally, we provide the security analysis for both schemes.

### 4.1 PKC Based Local Authentication

Public-key cryptography has been used extensively in data encryption, digital signature, user authentication, etc. Compared with the popular symmetric key cryptography widely used in sensor network, public-key cryptography provides a more flexible and simple interface requiring no complicated key pre-distribution and management as in symmetric-key schemes. It is a popular belief, however, in sensor network research community that public-key cryptography is not practical because the required computational intensity is not suitable for resource constrained sensor nodes. The nascent exploration seems to disabuse of the misconception. The recent progress in 160-bit Elliptic Curve Cryptography (ECC) implementation [9] on Atmel ATmega128, a CPU of 8Hz and 8 bits, shows that an ECC point multiplication takes less than one second, which proves public-key cryptography is feasible for sensor network security related applications.

We present our ECC based local authentication scheme as follows. $KDC$ selects a particular elliptic curve over a finite field $GF(p)$ (where $p$ is a prime), and publishes base point $P$ with order $q$ (where $q$ is also a large prime). $KDC$ picks a random number $x \in GF(q)$ as the system private key, and publishes its corresponding public key $Q = x \times P$. Given point $P$ and $Q$, it is computationally infeasible to get system secret $x$.

A straightforward user authentication scheme can be described as follows. The user uses her private key to sign her access list and sends to the sensors. The sensors just verify the signature by using user's public key. However, it is difficult for the sensors to find an authorized third party to certify that the user is who she claims to be. To solve this problem, we adopt the certificate-based authentication in our local authentication scheme. To access the sensor network, the user has to present her certificate first. Based on the certificate, the sensors generate user's public key, and then use the derived public key to encrypt a random number as the challenge. If the user can successfully decrypt the message, then the *local* sensors are convinced that the user's certificate is legitimate.

Initially, the user comes to KDC to apply for an access list to visit the sensor network. $KDC$ picks a random number $c_A \in GF(p)$, and then calculates the

user's public key constructor $C_A = c_A \times P$. Based on the user's request, $KDC$ issues a proper access control list $ac_A$, and attaches it to public constructor $C_A$ as the certificate, denoted as $T_A$. Meanwhile, a digest $e_A$ is generated for $T_A$, where $e_A = H(T_A)$ ($H$ is a $\{0,1\}^* \rightarrow \{0,1\}^q$ hash function). Then, $KDC$ constructs Alice's private key $q_A = e_A c_A + x$ and public key $Q_A = e_A \times C_A + Q$. Note $q_A$ and $Q_A$ satisfy $Q_A = q_A \times P$. Finally, Alice holds $q_A$, $Q_A$ and $T_A$. We assume above procedure is conducted at an out-of-band secure channel.

The user authentication protocol is illustrated in Fig. 2. We denote $s_l$ as a local sensor. When the user approaches a sensor node $s_l$, she sends her access request with certificate $T_A$. Given certificate $T_A$, $s_l$ constructs user's public key $Q_A = e_A \times C_A + Q$. To verify the user indeed holds private key $q_A$, node $s_l$ uses the challenge as follows. $s_l$ selects a random number $r \in GF(p)$ (to be used as the session key with the user), and calculate its hash $H(r)$ over GF(p). Node $s_l$ then generates temporary public key $Y_r = H(r) \times P$, and computes $Z_r = H(r) \times Q_A$. Next, $s_l$ encrypts the session key by doing $r \oplus X(Z_r)$, where $X(Z_r)$ is the X coordinate of point $Z_r$. Finally, $s_l$ sends ciphertext $\langle z_r, Y_r \rangle$ to the user, attached with the $MAC$ of a nonce ($N_A$), $MAC(r, N_A)$.

With private key $q_A$, the user can regenerate $Z_r$ because $q_A \times Y_r = q_A \times H(r) \times P = H(r) \times Q_A = Z_r$. She then decrypts session key $r = z_r \oplus X(Z_r)$, and verifies if $Y_r = H(r) \times P$. If yes, She uses $r$ as the session key to generate $MAC$ for nonce $N_A$ concatenated with her access privilege $ac_A$, and sends to $s_l$.

Local sensor $s_l$ decrypts the MAC message and verifies $N_A$ and $ac_A$. A successful verification proves that the user is the owner of certificate $T_A$. Finally, $s_l$ replies the information requested by the user, which again is encrypted by session key $r$.

$$
\begin{aligned}
user \rightarrow s_l &: T_A = (C_A | ac_A) \\
s_l \text{ computes} &: Q_A = e_A \times C_A + Q \\
&: \text{picks a random } r \in GF(p) \\
&: Z_r = H(r) \times Q_A, \\
&: Y_r = H(r) \times P, \\
&: z_r = r \oplus X(Z_r), \\
&: MAC(r, N_A). \\
s_l \rightarrow usesr &: z_r, Y_r, MAC(r, N_A) \\
user \text{ computes} &: q_A \times Y_r = q_A \times H(r) \times P = Z_r \\
&: X(Z_r) \oplus z_r = r \\
&: \text{decrypts } MAC(r, N_A) \\
user \rightarrow s_l &: MAC(r, N_A | ac_A) \\
s_l \rightarrow user &: MAC(r, reply)
\end{aligned}
$$

**Fig. 2.** User access list authentication protocol. We let $s_l$ be the local sensor, $T_A$ be the user certificate, which includes a public-key constructor $C_A$ and an access list $ac_A$.

### 4.2 Remote Access Control

In remote access control, the *remote* sensor node cannot directly contact the user due to the limitation of radio transmission range. Therefore, the user queries have to travel multiple hops to reach the *remote* sensor. With this communication pattern, the authentication schemes used in local access control cannot be applied on remote access control. In other words, it is improper for the user to directly contact the *remote* sensor. Otherwise the adversary can easily take the advantage and launch the bogus data injection attack to deplete the sensor network. With the above security concern in mind, we develop a remote access scheme that uses *local* sensors to endorse the user query to the *remote* sensor. Since it is widely accepted [11, 12] that a single sensor node cannot be trusted, the user's remote access request has to be endorsed by $k$ local sensor nodes, where $k$ is a system parameter. We assume the adversary cannot compromise $k$ sensors at a time. Any user remote access query without $k$ local endorsements will be dropped immediately by either forwarding sensor nodes or the *remote* sensor. A caveat is that some sensors may be compromised if a valid user cannot be authenticated by a group of sensors. In that case, the user can move to find another group of sensors for authentication or report the failure to the base station for analysis.

The requirement of *local* sensor endorsement raises a new security challenge: how does the remote sensor verify that the user is indeed endorsed by $k$ *local* sensors? If each local endorsing sensor can share a secret with the *remote* sensor, then the endorsement can be easily verified by the *remote* sensor. We use polynomial-based scheme for secret sharing between the local and remote sensors. More specifically, the KDC randomly generates a bivariate t-degree polynomial $f(x, y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j$ over a finite field $GF(q)$, where $q$ is a prime number and $a_{ij} = a_{ji}$. The polynomial has the symmetric property such that $f(x, y) = f(y, x)$. In practice, we select $t = k - 1$ so that the polynomial can not be reconstructed by the adversary with the assumption that the adversary cannot compromise up to $k$ sensors. To endorse a user access list, each local sensor can encrypt the access list with the key shared with the remote sensor, computed by substituting $x$ and $y$ with the sensor IDs. This scheme, however, has to provide the remote sensor with the IDs of the local sensors for verification, which leads to a long message. In order to reduce the message size, before the deployment, sensor nodes are divided into $k$ groups $\{g_1, g_2, \cdots, g_k\}$, where $g_j$ $(1 \leq j \leq k)$ is a group ID. Besides the group ID, each sensor $i$ has its unique sensor ID $s_i$. From now on, we also denote a sensor node as $s_i^j$, where $s_i$ is the sensor ID, and $j$ means it is belong to group $g_j$. During configuration procedure, each sensor $s_i^j$ is pre-loaded with two shares of polynomial, $f(x, s_i)$ and $f(x, g_j)$. Given the *remote* sensor ID $s_r$, a local sensor $s_{i_1}^{j_1}$ can establish a pairwise key with the *remote* sensor by plugging $s_r^{j_r}$ in $f(x, g_{j_1})$. And, the remote sensor can also generate the pairwise key by plugging group ID $g_{j_1}$ in its $f(x, s_r)$. To use group ID instead of sensor ID, we can achieve a shorter message due to a small number of groups. For the remote sensor to check the authentication list, we attach a bitmap for the groups in the message showing which group IDs are

user finds k local sensors $s_i^j$ with different j
$user \rightarrow s_1, \cdots, s_{k'} (k' \geq k)$ : bcast. request
$s_1, \cdots, s_{k'} \rightarrow user$ : group id
$user \rightarrow s_{p_1}, \cdots, s_{p_k}$ : confirm request
for (each sensor $\boldsymbol{s_{p_i}^{g_i}, i = 1, 2, \cdots, k}$)
$s_{p_i}^{g_i}$ authenticate user access list $T_A$
$s_{p_i}^{g_i} \rightarrow user : mac_i = MAC(f(s_r, g_i), ac_A)$
user computes: $mac = H(mac_1 || \cdots || mac_k)$
$user \rightarrow s_r : MAC(mac, ac_A || N_A) || ac_A ||$ group list
$s_r$ : compute $f(g_1, s_r), \cdots, f(g_k, s_r)$
$s_r$ : reconstruct $mac = Hash(mac_1 || \cdots || mac_k)$
$s_r$ : decrypt and verify $ac_A$
$s_r \rightarrow user : MAC(mac, reply || N_A || N_B)$

**Fig. 3.** The polynomial based remote access control protocol

used for authentication. We incorporate the remote sensor ID in the polynomial computation rather than the group ID of the remote sensor to avoid the attack due to the scenario that a compromised sensor has the same group ID with the remote sensor and then can decode the shared keys between the local sensors and the remote sensor.

The remote access control protocol is described in Fig. 3. To start a remote access procedure, the user has to find $k$ endorsing sensors $s_i^j$ such that no two sensors have the same group ID. The user first broadcasts the remote access request, and the local sensors receiving the request reply with their group ID. The user then select $k$ local sensors with different group ID to form an endorsing sensor group. Note the user may have to broadcast the request several times due to the possible transmission collisions. Then, each endorsing sensor conducts the local authentication as described previously. After the user has been authenticated, sensor $s_i^j$ computes the pairwise key $f(s_r, g_i)$ with the remote sensor, and uses the key to encrypt user's access list $ac_A$. Note only the access list part of certificate $T_A$ is encrypted because the *remote* sensor does not need user's public key constructor $C_A$. The user collects $k$ MACs from the endorsing sensors and generates a hash digest, $mac = H(mac_1 || \cdots || mac_k)$, where $g_1 < g_2 < \cdots < g_k$.

After computing the hash digest, the user encrypts her access list $ac_A$ and $N_A$ with $mac$. Again, $N_A$ is a nonce to guarantee the message freshness. Then, the user sends it along with her access list $ac_A$ and the local endorsing sensor group list, to the *remote* sensor.

When a *remote* sensor (denoted as $s_r$) receives the access request from the user, $s_r$ retrieves the information in the group list and user access list to reconstruct the MAC digest as shown in the protocol, and then decrypts the user's access list $ac_A$. If the decrypted access list matches the one provided by the

user, it proves that the user has already been authenticated by $k$ local sensors. Sensor $s_r$ replies the user with the requested information, along with nonce $N_B$ randomly picked by $s_r$. Again, all data is encrypted by $mac$.

### 4.3 Security Analysis

In both access control schemes, the authentication messages are encrypted by MAC algorithm in the access control protocol, except the user certificate. As long as the MAC algorithm is secure (such as RC5[13]), and the secret key is large enough (at least 64 bits), any number of compromised sensors cannot break the ciphertext in the messages.

In the local authentication, the sensor nodes can not capture any secret from the user, nor can the user gain more access privilege than granted due to the nice security features of public-key cryptography. The 160-bit elliptic-curve crypto-system is considered to have the same security level as 1024-bit RSA. Given an elliptic curve $E$ over finite field $F$, to find system secret $x$ from the relation $Q = xP$ (where $P, Q$ are published system parameters) is equivalent to solve the discrete logarithm problem, which is considered computationally infeasible. During the local authentication procedure, user's certificate $T_A$ including access list $ca_A$ is transmitted in plaintext. The malicious sensors may duplicate the user certificate, or the adversary may capture the certificate by eavesdropping. The certificate information, however, can not help the adversary to impersonate the user and get the data service. The reason is that the local sensors use user's public key to encrypt the challenge (random number $r$). It is easy for the adversary to calculate the public key given the stolen certificate, but it is computationally infeasible to acquire the associated private key. As the result, the adversary is not able to correctly respond the challenge, so her access request will be rejected by any local sensor. Due to the same reason, the user cannot forge or alter her access list to acquire higher level access privileges or to extend the allowed access time period. Otherwise, the user will not be able to decrypt the challenge message from the local sensor because she does not have the private key associated to the certificate she claims. More importantly, the certificate-based local authentication effectively defends against user collusion attacks. The collusion among any number of users does not jeopardize the system secret for the reason explained above.

The security features of our remote access scheme lie on the local sensor group endorsement. The combination of our local endorsement scheme with existing false report filtering schemes, such as SEF [17] and IHA [14], can effectively prevent the potential DoS attacks. In our scheme, users are not allowed to send requests directly to the remote sensor. Any remote access request has to be endorsed by $k$ local sensors. Since the adversary can not compromise up to $k$ sensors (the system assumption), there is no way for an illegitimate user to get $k$ genuine MACs to access the remote sensor. If the adversary attempts to forge $k$ MACs, the bogus request will be immediately dropped by forwarding sensors in false report filtering. Again, the user still can not alter or forge her

access list in the remote access request. The local endorsing sensors generate the MACs using authenticated user access list. If the user forges her access list in the remote access request, the MAC verification at the remote sensor will fail, and the remote request will be rejected.

## 5 Experimental Results

To evaluate the proposed access control schemes, we have implemented both local access control and remote access control scheme on TelosB (TPR2420) motes, the latest research oriented mote developed by UC Berkeley. TelosB is powered by MSP430 microcontroller. MSP430 incorporates an 8MHz, 16-bit RISC CPU, 48K bytes flash memory (ROM) and 10K RAM. The RF transceiver on TelosB is IEEE 802.15.4/ZigBee compliant, and can have 250kbps data rate. To simplify the experiments, we have implemented the user module on TelosB motes instead of PDAs.

### 5.1 Metrics and Methodology

We use four metrics: authentication time, computation cost, communication cost, and power consumption, to evaluate the performance of access control protocols. The authentication time measures user perceived waiting time from sending out the access request to receiving the authentication confirmation. Computation cost is the amount of energy consumed in data processing. Similarly, communication cost is the energy used by RF transceiver. The power consumption is the total amount of energy used by all participating sensor nodes to assist one user access request.

**Table 1.** The amount of current draw on different operations for TelosB motes

| Operation | Normal | Max |
|---|---|---|
| MCU On, Radio Off | $1.8mA$ | $2.4mA$ |
| MCU On, Radio Rx | $21.8mA$ | $23mA$ |
| MCU On, Radio Tx | $19.5mA$ | $21mA$ |

The energy consumption $E$ can be calculated by $E = U \cdot I \cdot t$, where $U$ is the voltage, $I$ is the current and $t$ is the time duration. TelosB motes are powered by two AA batteries, so $U$ is approximately equal to 3 volts. The current value varies in different operations as shown in Table 1 (abstracted from [4]). We use authentication time as the time duration for MCU data processing. And communication time can be estimated by following way. Given 250kbps radio transmission rate, and 38 bytes in each packet, it takes one sensor node $38 \times 8bits/250kbps = 1.2ms$ to transmit or receive a data packet. Without considering message loss and retransmission, the total transmission time is the product of $1.2ms$ with the number of packets.

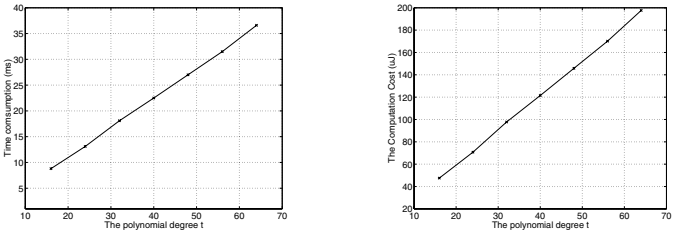
### 5.2 Experiment of Local Access Control

We have implemented 160-bit ECC cryptosystem on TelosB motes. We choose SECG recommended 160-bit elliptic curve, secp160r1, in our ECC implementation because large integer multiplication and reduction over prime number finite field can be more effectively optimized than those over binary finite field. The most expensive operation in ECC exponentiation is point multiplication. To achieve the better performance as possible, we have adopted a number of techniques including hybrid multiplication, modular reduction over pseudo-Mersenne prime field, Great Division and mixed Jacobian Coordinate. Due to the space limit, we omit the detail implementation and corresponding optimization of our ECC implementation on TelosB motes. Interested readers may refer to [16] for detail explanation. On average, it takes 3.1 seconds for a TelosB sensor mote to do a fixed point multiplication, and 3.5 seconds to do a random point multiplication. Note this performance is achieved under the circumstance that TelosB micro-controller's hardware multiplier is disabled in TinyOS.

Our local access control implementation strictly follows the protocol presented in section IV except that the data encryption/decryption part is not implemented due to the reason that TinySec (which provides block-cipher module) does not work with CC2420 radio module on TelosB, but it does not affect our performance evaluation because encryption/decryption overhead is negligible (e.g., in RC5, the most expensive step (key setup) only costs 4ms on ATmega128 [8]) compared with ECC exponentiation.

The user certificate $T_A$ has 48 bytes, including 40-byte public key constructor and 8-byte access list. The challenge from sensor nodes has 80 bytes, including a 40 byte ECC point, 20 byte $z_r$ and a 20 byte ciphertext. Since one TelosB packet only has 28 byte payload, the user has to use multiple packets to deliver the certificate. In total, user needs to send four messages (three messages to deliver user certificate, the forth one to response sensor's challenge). Similarly, the local sensor also needs to send four messages to deliver the challenge. We use challenge generation time as our authentication delay. The challenge generation time is user perceived delay from sending out the access request to receive the challenge from the sensor. We exclude the user response time from the authenticate delay because the user usually carry much more powerful devices in the real world, so the response time is negligible compared with sensor processing time.

Our experiment results show that a challenge generation costs 10.8 seconds on average. Obviously, computation delay dominates communication delay in this procedure. Recall that a sensor node needs to perform two ECC random point multiplications and one fixed point multiplication to generate a challenge. The three point multiplications combined already contribute 10.1 second delay. The communication delay to send/receive 8 packets only has $8 \times 1.2ms = 9.6$ milliseconds. The power consumption for the computation is 58.3mJ, while the energy cost for the communication is 0.59mJ.




**(a)** **(b)**

**Fig. 4.** (a). The time consumption to generate a pairwise key from the polynomial. (b). The power consumption to generate a pairwise key.

### 5.3 Experiment of Remote Access Control

The essential part of the experiment of remote access control is the polynomial based local endorsement scheme and MAC recovery at the remote sensor. We are particularly interested in the performance of the t-degree polynomial computation in sensors. Given a share of the polynomial $f(x) = a_0 + a_1 x + \cdots + a_t x^t$ over $GF(q)$, the computation of $f(x)$ requires $t$ modular multiplications and $t$ modular additions, plus the computation of values $x^2, \cdots, x^t$. A typical cryptosystem (e.g., RC5) suggests $q$ should be at least 64 bits. Therefore, $t$ 64-bit $\times$ 64-bit modular multiplications are required to compute the polynomial. On TelosB's 16-bit CPU platform, each 64-bit $\times$ 64-bit multiplication costs 16 word multiplications. To reduce the computational cost, we adopt the simplification proposed in [10]. The simplification is based on the fact that variable $x$ is either sensor ID or group ID, which is normally a 16-bit integer. We can use another finite field $GF(q')$ for $x, x^2, \cdots, x^t$. Therefore, the modular multiplication in polynomial $f(x)$ is always performed between a 64-bit integer and 16-bit integer. As the result, the cost of multiplication is reduced by four times.

The modular reduction operation is as important as multiplication. Each multiplication must be followed by a reduction operation. To further reduce the computational cost, we pick a pseudo-Mersenne prime as $q$ because modular reduction cost on field of a pseudo-Mersenne prime can be optimized to a negligible amount. A pseudo-Mersenne prime can be represented as $q = 2^m - \omega$, where $\omega << 2^m$. Given a $2m$-bit multiplication result $B = (b_1, b_0)$, ($b_1, b_0$ are two $m$-bit halves), the reduction can be computed based on the congruence $2^m \equiv \omega$: $(b_1, b_0) = b_1 * \omega + b_0 \rightarrow (b_1', b_0')$. Repeat this process until $b_1' = 0$, the result is $B = b_0' \bmod q$.

In our experiment, we choose $q = 2^{64} - 2^8 - 1$, $q' = 2^{16} - 2^4 - 1$. We test the average time delay and power consumption for computing the polynomial with different $t$ values. In each test, we randomly generate $t+1$ 64-bit coefficients and a 16-bit variable $x$, we repeat 20 times to get the average time delay. The test results are shown in Fig. 4.

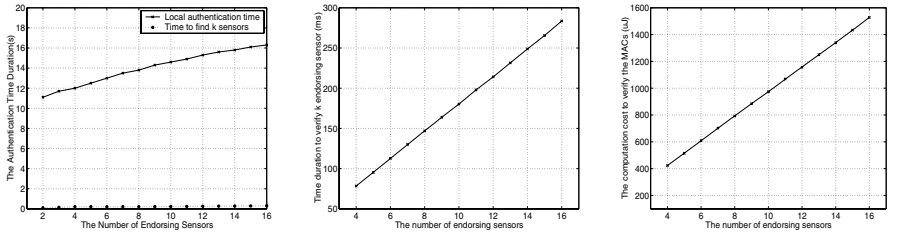
The test results show the polynomial computation is efficient in low-power sensor nodes. The figure shows that the time consumption for generating a pair-

wise key is only 8.8ms, 17.1ms, and 36.8ms, given the polynomial degree of 16, 32, and 64, respectively.

To evaluate the remote access control procedure, we divide the experiment into two parts. The first part includes local sensor discovery, local sensor authentication and MAC collecting. In the second part, we perform the MAC reconstruction and verification at the remote sensor. The message routing between the user and the remote sensor is a typical communication process that has been investigated extensively and the time delay is very small, so in our experiment we omit the message routing between the user and the remote sensor.

During the experiment, we assume the sensor field is dense enough so that the user can reach *local* sensors from different groups without moving. To acquire the endorsements from local sensors, the user first broadcasts a remote access request. Each local sensor replies the user with its group ID. The user picks those sensors from different groups to fill in her endorse list. Due to the message collision, some replying messages are corrupted, so the user may not find enough endorsing sensors with one broadcast. As the result, the user may have to broadcast several times to find all $k$ endorsing nodes. Our experiments show the user has to broadcast at least twice if $k \geq 6$. After successfully finding $k$ endorsing sensors, the user unicasts an endorse acknowledge to each of the $k$ sensors. The endorsing sensors processes the user authentication in parallel. The user first broadcasts her certificate, and then sequentially receives and responses each local sensor's challenge. A simple scheduling algorithm can be used for the endorsing sensors to send challenges without packet collision. In our implementation, we arrange the endorsing sensors to send the challenge in ascending order of their group IDs. If the user is successfully authenticated, then each endorsing sensor generates the MAC and returns it to the user. After collecting all $k$ MACs from endorsing sensors, the user finally generates a MAC digest and sends the access request to the remote sensor. We perform the experiment with $k$ changing from 2 to 16. The result of endorsing time consumption is shown in Fig. 5(a). Note that the time duration includes the time for user's broadcasts for request, receiving the group ID reply from sensors, unicasts to sensors for acknowledging receiving their group IDs, and sensor nodes' data processing time to generate the MACs.

We first perform a separate experiment just to test the time delay to find $k$ sensors only (without local authentication and MAC generation). The result is shown as the dotted line in the same Fig. 5(a). It is interesting to find that it takes $105ms$ to find just 2 endorsing sensors and considerable time for discovering 4, 8, and 16 sensors, which is surprisingly slow, considering $1ms$ transmitting/receiving delay. Two factors contribute to the long delay. First, as discussed in previous section, the user may not get all information from local endorsing sensors after the first broadcast. The user may have to broadcast the request more than twice. Second, more importantly, a timer is set between any two broadcasts in our implementation to regulate the packet transmission and reception. Every time the timer fires, the user checks whether the endorsing list is complete. If not complete, the user will do broadcast again. The time delay




**(a)** **(b)** **(c)**

**Fig. 5.** (a). The solid line shows the time duration for the user to get authenticated by $k$ local sensor, $k$ is changing from 1 to 16. The dotted line reveals the time delay for the user to find $k$ endorsing sensors; (b). The time duration for remote sensor to verify $k$ endorsing local sensors; (c). The energy cost for the remote sensor to verify $k$ endorsing local sensors.

between the fires of the timer predominantly accounts for the sensor discovery delay. We can reduce this time duration by setting a higher timer frequency.

The total endorsing time is presented in Fig. 5(a) (solid line). Apparently, the expensive local authentication dominates other delays. However, because $k$ local sensor authenticate the user in parallel, the total endorsing time is practical and not much longer than the local authentication delay. When $k = 16$, it only takes 16.7 seconds for the user to get all endorsements.

Once receiving user's remote access request, the remote sensor has to verify whether the user is endorsed by $k$ local sensors. To do so, the remote sensor reconstructs $k$ MACs by plugging the group ID into its own share of polynomial. After $k$ MACs are reconstructed, the remote sensor then generates and verifies the digest. In the experiment, we measure the time duration for the remote sensor to do the verification with $k = 4, 5, \cdots, 16$ endorsing sensors. The experiment results are shown in Fig. 5(b)(c).

Finally, we estimate the total time for a user to be authenticated for remote data access. Suppose the network requires the user to get 16 endorsing sensors to access a remote sensor. First, the user has to get local authentication by all 16 local sensors and receive corresponding MACs. This procedure costs 16.7 seconds according to Fig. 5(a). Then, the remote sensor needs $283ms$ to reconstruct and verify 16 MACs. In total, a remote access with 16 local sensor endorsement will cost around 17 seconds. Note that our estimation does not include the message traveling time from the user to the remote sensor and then back to the user.

## 6 Conclusion

In this paper, we show our effort in designing access control scheme for sensor networks. We describe our local access control and remote access control under a very realistic adversary model. We implement the protocols on a TelosB mote testbed. The security and performance analysis and the experimental results show that our access control is feasible for real application. We are currently in the process of doing more experiments and designing more schemes for access control for comparison.

## References


1. D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong. Secret handshakes from pairing-based key agreements. In *2003 IEEE Symposium on Security and Privacy*, Berkeley, CA, May 2003.
2. H. Chan and A. Perrig. Pike: Peer intermediaries for key establishment in sensor networks. In *INFOCOM 2005*, Miami, FL, March 2005.
3. H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *In IEEE Symposium on Security and Privacy*, pages 197–213, Berkeley, California, May 2003.
4. Moteiv Co. Telos datasheet. *http://www.moteiv.com /products/docs /tmote-sky-datasheet.pdf*.
5. W. Du and J. Deng. A pairwise key pre-distribution scheme for wireless sensor networks. In *ACM CCS 2003*, 2003.
6. Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen, and Pramod Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE INFOCOM'04*, Hong Kong, March 2004.
7. L. Eschenauer and V.D. Gligor. A key-management scheme for distributed sensor networks. In *In Proceedings of the 9th ACM conference on Computer and Communication Security*, November 2002.
8. Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *WSNA03*, San Diego, CA, Sept 2003.
9. Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *CHES*, Boston, Aug. 2004.
10. D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS'03*, Washington, DC, October 2003.
11. A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of The ACM*, 47(6):53–57, June 2004.
12. A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. Spins: Security protocols for sensor networks. *ACM/Kluwer Wireless Networks Journal (WINET)*, September 2002.
13. Ronald L. Rivest. The rc5 encryption algorithm. In *Proceedings of the 1994 Leuven Workshop on Fast Software Encryption (Springer 1995)*, pages 86–96, Springer, 1995.
14. S. Jajodia S. Zhu, S. Setia and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *In Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.
15. Harald Vogt. Exploring message authentication in sensor networks. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, Heidelberg, Germany, August 2004.
16. H. Wang, B. Sheng, and Q. Li. Telosb implementation of elliptic curve cryptography over primary field. In *Technical Report*, Dec 2005.
17. F. Ye, H. Luo, S. Lu, and L. Zhang. Statistical en-route filtering of injected false data in sensor networks. In *INFOCOM 2004*, 2004.
18. W. Zhang, H. Song, S. Zhu, and G. Cao. Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks. In *MobiHoc'05*, Chicago, IL, May 2005.