

Efficient and Distributed Access Control for Sensor Networks

Donggang Liu

iSec laboratory, CSE Department
The University of Texas at Arlington
dliu@uta.edu

Abstract. Sensor networks are often used to sense the physical world and provide observations for various uses. In hostile environments, it is critical to control the network access to ensure the integrity, availability, and at times confidentiality of the sensor data. This paper develops efficient methods for distributed access control in sensor networks. The paper starts with a *baseline approach*, which provides a more flexible and efficient way to enforce access control when compared with previous solutions. This paper then extends the baseline approach to enable *privilege delegation*, which allows a user to delegate its privilege to other users without using a trusted server, and *broadcast query*, which allows a user to access the network at a large scale efficiently. The privilege delegation and broadcast query are very useful in practice; none of the current solutions can achieve these two properties.

1 Introduction

The primary purpose of deploying a sensor network is to monitor the physical world and provide observations for various uses. In hostile environments, it is critical to control the access to the sensor nodes (e.g., reading sensor data), especially when there are many users in the system. For example, a sensor network may be deployed to monitor the activities in a battlefield and provide information for soldiers and commanders to make decisions during military operations. Every soldier or commander will be a potential system user. In applications like this, different users may have different access privileges. A soldier may be only allowed to read the sensor data, while a commander may be able to re-configure the network or update the internal states of the sensor nodes. The application will be compromised if the access control is not properly enforced.

However, enforcing access control in sensor networks is particularly challenging. First, the resource constraints on sensor nodes often make it undesirable to implement expensive algorithms. For example, the commonly used MICAz platform uses an 8-bits ATmega128 CPU that operates at 7.7MHz [1]. It only has 128KB ROM for programming code and 4KB RAM for buffers and variables. Second, sensor nodes are usually deployed unattended and may be compromised after deployment [2]. Hence, any security protocol has to be resilient to node compromises.

A number of techniques have been developed recently to achieve access control in sensor networks. These include the *Least-Privilege* scheme [3] and the *Wang-Li* scheme [4]. However, the former can only be used for a specific type of access control where the access is limited to the data at a pre-determined physical path in the field. The latter is based on Elliptic Curve Cryptography (ECC), which has been shown to be feasible for sensor nodes [5]. However, the DoS attacks on signature verification were not properly addressed, making this scheme less attractive. In addition, this scheme can only access one sensor node at a time, while a significant part of actual access requests are targeted to many sensor nodes via broadcast.

In this paper, we start with a baseline approach to deal with the situation where every access request is targeted to a specific sensor node. Compared with the previous schemes, our baseline approach has a number of advantages. First, it provides a generic access control method and can be easily used to enforce any type of access control policies. Second, it only uses symmetric key cryptography, which is much more efficient than public key cryptography. Third, our approach only involves a single message for every request from the user to a sensor node, while the scheme in [4] needs to employ a more expensive challenge-responsive protocol for accessing a local sensor node and a much more complicated cooperative protocol for accessing a remote sensor node. Finally, our approach provides strong security guarantee against compromised sensor nodes or users. Indeed, no matter how many sensor nodes or users are compromised, a benign sensor node will not grant the attacker any access that is beyond the simple union of the privileges of the compromised users.

In addition to the baseline approach, we also develop techniques that enable *privilege delegation* and *broadcast query*. The idea of privilege delegation allows a user to delegate its access privilege to other users without a trusted third party. The idea of broadcast query allows a user to efficiently access the sensor nodes at a large scale. These two additional functionalities can be very useful in many sensor operations, making our access control approach much more appealing than the previous solutions.

This paper is organized as follows. The next section gives the system model and assumptions. Section 3 presents the proposed techniques for access control in sensor networks. Section 4 shows the evaluation of the proposed techniques. Section 5 reviews some related work on sensor network security. The last section summarizes this paper and discusses future research directions.

2 System Models and Assumptions

In this paper, we consider wireless sensor networks that consist of a large number of resource-constrained sensor nodes, many system users, and a trusted server such as the base station. The sensor nodes are used to sense conditions in their local surroundings and report their observations to system users based on various query commands. The system users (e.g., soldiers) use *access devices* such as PDAs and Laptops to access the sensor data. The trusted server is used to bootstrap the keying materials for access devices to enforce access control policy.

Dataset Table: For every sensor node, the sensor data that are accessible for system users can be viewed as a *dataset table*. Every record in this table is a snapshot of the values of different *attributes* at a particular time. An attribute is defined as a specific type of sensor data in the network. Examples of attributes include the sensor readings (e.g., temperature) and the sensor states (e.g., the active sensor). Table 1 shows an example of the dataset table of a sensor node at time t_n . This node is able to sense the temperature and humidity of its local environments. It may activate one or both sensors, depending on the value of the third attribute, the active sensor.

Let $\mathcal{A} = \{a_1, a_2, \dots, a_m\}$ be the set of attributes in the system. Any value d in the dataset table can be located based on the attribute name $a \in \mathcal{A}$ and the time t . For simplicity, let $d_{a,t}$ be the value of the attribute a at time t , $A(d)$ be the attribute name related to data d , and $T(d)$ be the time related to data d . Clearly, $A(d_{a,t}) = a$ and $T(d_{a,t}) = t$.

Table 1. Dataset table at time t_n

Time	Temp.	Hum.	Active Sensor
t_1	80	N/A	temp. sensor
t_2	82	N/A	temp. sensor
t_3	N/A	45	hum. sensor
\vdots	\vdots	\vdots	\vdots
t_n	N/A	50	hum. sensor

Access Model: A system user can perform two possible access operations on a particular data value in the table, *read* and *write*. For example, the sensor readings such as temperature and humidity are usually read only, while the attributes related to the system states can be modified by a user with some proper privilege. Based on the dataset table, we define a *capability matrix* for a given user. A two-bit value at column i and row j in this matrix indicates the user’s access privilege to d_{a_i,t_j} in the dataset table. This value can be 00 (no access), 01 (read-only), 10 (write-only), or 11 (read and write).

Note that the capability matrix can be very large. Fortunately, most sensor applications use a simple way to specify a user’s capability. For example, the policy “user U can only read the humidity readings collected from time t_1 to time t_2 ” can be encoded as “ $(A(d) = \textit{humidity}) \wedge (t_1 \leq T(d) \leq t_2) \wedge (O = 01)$ ”, where O denotes the operation performed by the user. Hence, we can usually encode the capability matrix into a short *capability string*. We will not discuss the details of the encoding and decoding schemes for capability strings since they are application-specific and orthogonal to our approaches.

A user’s capability string can be used to define its access privilege. However, we might have other requirements for the user’s access to the sensor data. For example, the user may only be able to access the data at a rate of no more than one packet per 30 seconds. Hence, we define a user’s access privilege as a **constraint**. The user’s capability string can be also viewed as a constraint. A constraint can be built from multiple constraints by simply using “AND” (\wedge) or “OR” (\vee) operator. For example, “ $(A(d) = \textit{humidity} \wedge O = 01) \wedge (R \geq 30)$ ” means that the user can only read the sensed humidity at a rate of no more than one packet per 30 seconds. In this constraint, the first part is the capability

string of the user, while the second part ($R \geq 30$) is an additional constraint on the packet rate.

A constraint can usually be written in an appropriate formal language. However, our method is independent from how we write the constraints. Hence, we simply discuss it informally rather than proposing a formal language to define constraints. As a result, we denote the access privilege of a user u as a constraint C_u . The access will be granted to u only when the queried sensor data meets the constraint C_u . This access model is simple yet flexible for a sensor network. We believe that it is sufficient for most sensing applications.

Attack Model: An attacker can launch a wide range of attacks against the network. For example, he can simply perform a denial-of-service (DoS) attack to jam the wireless channel and disable the network operation. Such DoS attack is easy to mount and common to the protocols in every sensor network. There are no effective ways to stop the attacker from doing such attack. However, in this paper, *we are more interested in the attacks whose goal is to access the valuable sensor data that he is not supposed to access.* We assume an attacker can eavesdrop, modify, forge, replay or block any network traffic. We assume that it is computationally infeasible to break the underlying cryptographic primitives such as encryption, decryption, and hash. We assume that the attacker is able to compromise a few sensor nodes and learn the key materials stored on the compromised sensor nodes. We also assume that some users may be compromised. Once a user is compromised, all the secrets assigned to the user will be disclosed.

3 Efficient and Distributed Access Control

This section presents the proposed method for access control as well as the detailed analysis. We start with a baseline approach and then develop techniques to enable *privilege delegation* and *broadcast query*.

3.1 The Baseline Approach

The main focus of the baseline approach is to provide efficient access control when a user queries only one node at a time. With this technique, it will be also practical to access a few nodes at a time as long as the query packet has space for additional information such as MACs. We assume that the user knows the ID of the sensor that has the requested data. We also assume a routing protocol that can be used to deliver the query to such node.

It is clearly possible that the query message gets lost on the way to the queried node due to the unreliable channel or malicious attacks. For example, an attacker can always jam the channel and interrupt the delivery of any message. There are no effective methods to prevent an attacker from mounting such DoS attacks. Therefore, our focus in this paper is to stop unauthorized user access to the sensor networks.

The main idea of our baseline approach is to map the access privilege of a user into a cryptographic key. This key is used to prove that the user does have the access privilege he claims. As a result, achieving confidentiality and integrity using this key also achieves the access control.

- *Initialization:* Before deployment, every node i shares a key K_i with a trusted server. Every user u will contact this server for the keying materials for access control. The server first determines the constraint C_u based on the user’s credentials, depending on the application. With the constraint C_u , the user u gets pre-loaded a hash key $K_{u,i} = H(u||C_u||K_i)$ for every node i , where H is a one-way hash function and “||” denotes the concatenation of two bit strings. Clearly, for a network of n nodes, a user will need to store n hash keys. We believe that this storage overhead will not be a problem for users who have powerful computing devices such as PDAs or Laptops. In fact, even for a resource-poor sensor node such as TelosB motes [1], it has 1MB flash memory to store the keying materials for a network of 128,000 nodes if every hash image is 8-byte long.
- *Access Query:* Let $\{\cdot\}_K$ be the message authentication operation using K . When a user u wants to access a sensor node i , it constructs an appropriate query command $Q(u)$ and send the following message to node i .

$$u \rightarrow i : \{C_u, Q(u)\}_{K_{u,i}}$$

Once node i receives such message, it can re-construct the hash key $K_{u,i}$ from C_u since it has the key K_i . With this hash key, it can check the authenticity of the query message. If the message is authenticated, it is certain that user u does have the access privilege defined by C_u and the query $Q(u)$ does come from u . If $Q(u)$ also meets the constraint C_u , the access to i ’s data defined by $Q(u)$ will be granted to u .

Security: The focus of our security study is *the access to the data at non-compromised nodes*. For a benign user u , we can see that a benign node i will provide access to this user only when u does make a corresponding request. The reason is that every request from user u to node i has to be authenticated by the key $K_{u,i}$, which is only known by user u , node i and the trusted server.

We then study the access capability of an adversary who compromised a set of system users U_c . Assume that an adversary issues a query command Q_c to access the sensor data. The following theorem tells us that no matter how many sensor nodes are compromised, the collusion of compromised users and sensor nodes will not give the adversary any additional privilege that is beyond the simple union of privileges of compromised users.

Theorem 1. *To access the sensor data at benign sensor nodes, the query Q_c from the attacker must meet the constraint $\bigvee_{v \in U_c} (C_v)$, where U_c is the set of compromised users.*

Proof. According to the initialization step, every keying material at a user is generated using a one-way hash function. Hence, no matter how many sensor nodes or system users are compromised, an adversary is unable to access the communication between a benign user and a benign sensor node. Hence, the adversary must issue Q_c as a user $v \in U_c$ since otherwise Q_c will be immediately rejected because of the failure in the authentication. When $v \in U_c$, if the query Q_c doesn’t meet the constraint C_v , Q_c will also be rejected by any benign sensor

node i . Hence, the query Q_c must meet $\bigvee_{v \in U_c} (C_v)$ in order to gain access to a benign sensor node.

Overheads: In the baseline approach, a sensor node only needs to store a single key, while a system user needs to store n keys for a network of n nodes. However, as we mentioned, this is usually not a problem for the users who have powerful and resourceful platforms such as PDAs and Laptops. To access the data at a sensor node, a user only needs to send a single message. For every query message, the user only needs to perform one efficient hash operation, while the target node only needs to do two efficient hash operations, one for generating the authentication key and the other for authentication.

Comparison: The baseline approach has many advantages over the previous schemes in [3, 4]. First, it provides a generic access control method and can be easily used to enforce any type of access policies. While the technique in [3] can only achieve a specific type of access control where the system can only force the user to access the network at a pre-determined physical path in the field. Second, our approach only involves a few efficient one-way hash operations. This is much more efficient than the ECC-based signature scheme in [4]. Third, our approach only involves a single message for each query, while the schemes in [4] need three messages for a local sensor node and a lot more messages for a remote sensor node. In their method, a certain number of local sensor nodes have to collaborate together to commit the query message for a remote sensor node. This complicates the protocol and increases the overheads significantly. Finally, our approach is more resilient to the compromise of sensor nodes. Indeed, no matter how many sensor nodes or users are compromised, the control of the access from any user to a benign sensor node can still be properly enforced. In contrast, the collusion of a small number of sensor nodes will allow the attacker to access any sensor node for the technique in [4].

3.2 Enabling Privilege Delegation

In many cases, a user may want to directly delegate its privilege to other users in the system without going through a trusted server. This can be very useful when the trusted server is not available for bootstrapping the keying materials. For example, in a battlefield, the trusted server may stay in the military base, while an officer may want to temporarily delegate its privilege to a soldier during the military operation in the field. In this case, it is often not practical to contact the trusted server to bootstrap the keying materials for the soldier.

Fortunately, privilege delegation can be easily implemented in our baseline approach. The overall process is similar to the bootstrap in the baseline approach. Basically, whenever a user wants to delegate its privilege to other users, it can act as a trusted server since it shares a unique key with every sensor node. A user with a delegated privilege can further delegate its privilege to other users, forming a *delegation tree*. For the sake of presentation, when a user u delegates its privilege to user v , we call u as v 's *parent user* and v as u 's *child user*. Figure 1 shows an example of delegation trees.

For every user u in the delegation tree, it knows the IDs as well as the constraints of the users who are on the path from u to the root (the trusted server). This information is pre-loaded to u when the trusted server or a user bootstraps u . For the sake of presentation, we use $S(u)$ and $I(u)$ to denote the set of constraints and IDs of the users on the path from u to the root in the tree respectively. For example, in Figure 1, we have $S(7) = \{C_7, C_6, C_2\}$ and $I(7) = \{7, 6, 2\}$.

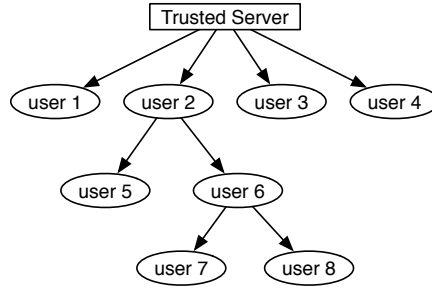


Fig. 1. An example of delegation tree

- *Delegation:* Assume a user u wants to delegate its privilege to another user v . u first determines the constraint C_v for user v . User u will then assign a hash key $K_{v,i} = H(v||C_v||K_{u,i})$ to user v for every sensor node i . User u also pre-loads $I(v) = I(u) \cup \{v\}$ and $S(v) = S(u) \cup \{C_v\}$ to v . The actual access privilege of user v will be determined by $S(v)$. For example, in Figure 1, the privilege of user 6 is $C_2 \wedge C_6$.
- *Access Query:* When user v needs to access node i , it sends the following message to i .

$$v \rightarrow i : \{I(v), S(v), Q(v)\}_{K_{v,i}}$$

After the node i receives such message, it can easily reconstruct $K_{v,i}$ based on $I(v)$ and $S(v)$. When the message is verified, it is certain that user v does have the access privilege defined by $S(v)$ and the query $Q(v)$ does come from v as long as the users on the path from v to the root of the delegation tree are still benign. Finally, node i will check if $Q(v)$ meets all the constraints in $S(v)$. If yes, the access is granted.

Privilege delegation may cause problems for *stateful* constraints. A user's constraint is said to be stateful if it requires the node to maintain certain history information for this user. A potential security problem for these constraints is that a malicious user can easily bypass them by privilege delegation. As one example, assume a malicious user can only collect data at a rate of no more than one packet per 30 seconds. To bypass this constraint, this user can simply create another user and delegate its privilege to the newly created user to initiate another information flow with up to one packet per 30 seconds.

To address this problem, we propose an *upward updating* approach. In this approach, when node i receives an authenticated query from user u , it will consider it as the valid query for every user $v \in I(u)$ and check with every $v \in I(u)$ to see if it meets all the constraints in $S(v)$. If not, the query command will be rejected; otherwise, the query will be accepted, and the sensor node i will update its state for every user $v \in I(u)$ based on the query as well as the response to such query.

Security: The delegation of a user’s privilege is similar to the process of bootstrapping the keying materials from the trusted server. Due to the one way property of the hash function H , we can see that an adversary who compromised a set U_c of users can only access the sensor data that are accessible for at least one user in U_c . Indeed, the access of adversary is still limited by the following constraint: $\bigvee_{u \in U_c} (S(u))$. The proof will be the same as the proof of Theorem 1. We will then skip the detail of the proof for simplicity.

Overheads: It is worth noting that providing the privilege delegation property will not increase the storage overhead at sensor nodes. The only difference is that the states of different users are now organized as a tree structure. As for the computational overhead, a user will need to generate n hash values to delegate its access privilege. A sensor node needs to perform a number of hash operations to generate the key to authenticate the query and protect the sensor data. This number depends on the distance from the corresponding user to the root of the delegation tree. Fortunately, the height of the delegation tree will not be very large in most cases. As for the communication overhead, we can clearly see that allowing privilege delegation will not introduce much more communication cost. A user only needs to send a single message, which has the similar format as the query message in the baseline approach, to access the data at a sensor node.

3.3 Enabling Efficient Broadcast Query

The baseline approach can only support the queries that are targeted to one sensor node at a time. In practice, however, a system user may want to access the data at many sensor nodes at a time. We can certainly use the baseline approach to query these nodes one by one. However, in this case, the communication overhead will increase linearly with the number of queried nodes. Hence, it is particularly desirable to enable *broadcast query*, where many sensor nodes can be queried at the same time efficiently.

A broadcast authentication method will be needed for broadcast query. Two possible candidates are μ TESLA [6] and the ECC-based signature scheme [5]. μ TESLA is based on symmetric cryptography and is believed to be more efficient than the ECC-based scheme. However, it has some undesirable features such as the need for time synchronization, the authentication delay, and the update of key chains. These limit the application of μ TESLA in real-world scenarios. On the other hand, the performance of ECC-based signature schemes have been and will continue to be optimized by researchers. We thus use ECC-based signature schemes for broadcast authentication in this paper. However, a critical problem that has to be addressed before making the ECC-based signature scheme a reality is the DoS attacks on signature verification. Hence, the main challenge is *how to thwart the DoS attacks against the signature verification*.

Note that broadcast is usually done through a network-wide flooding. Many energy efficient flood mechanisms could be used for this purpose [7–9]. Our main observation is that every flooding message from a sensor node only has a small number of receivers due to the limited neighbor size. This actually allows us to *weakly* authenticate the broadcast message before verifying the digital signature

to thwart the DoS attacks. Since our weak authentication method is independent from the broadcast protocol, we simply assume a broadcast protocol and will not discuss how it is achieved. In the following discussion, we assume that every two neighbor sensor nodes u and v share a pairwise key $k_{u,v}$. Many existing pairwise key establishment protocols could be used for this purpose [10–13]. Note that we use the lower-case “ k ” for the pairwise key between two sensor nodes and the upper-case “ K ” for the shared key between a user and a sensor node.

- *Initialization:* The initialization step is similar to the baseline approach. However, we have additional keying materials for sensor nodes and users. Specifically, the public key of the trusted server will be pre-loaded to every sensor node. Every user u also has a pair of private and public keys $(K_e(u), K_d(u))$. The public key $K_d(u)$ and C_u are either signed by the trusted server or another user who is willing to delegate its privilege to u . Let $Cert(u)$ be the resulting signature. In the end, user u will get pre-loaded with $Cert(v)$ for every $v \in I(u)$, where $I(u)$ is the set of user IDs on the path from u to the root of the delegation tree.

After deployment, every node i first discovers a set of neighbors $N(i)$ and exchange it with every node $j \in N(i)$. In the end, we want to make sure that a sensor node i has the neighbor list $N(i)$ and also knows its position $P_{j,i}$ in the set $N(j)$ for every $j \in N(i)$. Such position information is used to identify the values for weakly authenticating the broadcast message.

When a user u needs to query a large number of sensor nodes, it will first generate a broadcast message M_B that includes the constraint C_u , its public key $K_d(u)$, the certificate $Cert(u)$, the query command $Q(u)$, and a signature on $Q(u)$ using its private key. User u then sends M_B to one or a few randomly picked nearby nodes to start the broadcast query process.

- *Broadcast:* When a sensor node i gets an authenticated copy of M_B , it will check to see if it needs to re-broadcast the message based on the broadcast protocol. If yes, it will compute a set of *committing values* (one for each neighbor node) to *weakly authenticate* the broadcast message. Specifically, for each neighbor node j , node i computes $H(M_B || k_{i,j})$ and uses the most significant l bits of this hash as the $P_{i,j}$ -th committing value for weak authentication. Let W be the set of these committing values. The final broadcast message will be $\{M_B, W\}$.
- *Authentication:* When a sensor node j receives the message $\{M_B, W\}$ from node i for the first time, it computes $H(M_B || k_{i,j})$, extracts the most significant l bits, and then compares it with the value at the position $P_{i,j}$ in W . If it finds a different value, the packet will be ignored; otherwise, node j starts to perform two signature verifications on M_B , one for the certificate and the other for the signature signed by node i . Once both signatures are verified, it is certain that the privilege and the query of user u are correct. Node j will then grant access to u if the query meets the constraint. Similar to the baseline approach, the access will be protected by the key $K_{u,j}$. Note that when privilege delegation is enabled, node j will need to receive all the certificates of the users on the path from u to the root. This can be provided by u when it is necessary.

Note that a compromised or malicious node can forge the broadcast queries that can always pass the weak authentication and trigger the expensive signature verification at other sensor nodes. To deal with this problem, we develop an *anomaly suppression* method to make sure that a compromised sensor node cannot generate a large impact on the signature verification at other sensor nodes. The main observation is that *the fraction of forged broadcast messages that can pass weak authentication will be very small when the sender is a benign node*. Hence, if a sensor node observes a large fraction of forged messages passing weak authentication, the sender is very likely to be compromised.

- *Anomaly Suppression:* We keep track of the number x of forged broadcast messages that passed the weak authentication at node i during the previous M forged broadcast messages from node j . If $x > m$, node i will directly consider the message as forged and stop any further verification; otherwise, it will verify the certificate and signature once the broadcast message passes the weak authentication check. Clearly, the fraction of unnecessary signature verifications is bounded by $\frac{m}{M}$.

Security analysis: We study how well our approach performs under DoS attacks. Assume that node i receives a forged broadcast message $\{M'_B, W'\}$ from the adversary who impersonates node j . Note that the committing value for node i is generated using the key $k_{i,j}$ that is only known by nodes i and j . The probability that W' includes a correct guess of this value can be estimated by $\frac{1}{2^l}$. Clearly, our method can effectively reduce the impact of DoS attacks on signature verification even for a small l . For example, when $l = 8$, node i only needs to verify one out of 256 forged messages on average.

When node j is a compromised sensor node, the problem becomes much more complicated since node j can always forge broadcast messages with correct committing values. Fortunately, such impact is bounded by the threshold m and the window size M . Intuitively, setting a smaller m or a larger M brings more resilience against DoS attacks but generates more false suppressions for benign sensor nodes. Hence, we study the *false suppression rate*, which is *the probability that the number of forged messages that passed the weak authentication in a window of M fake messages exceeds m when the sender is actually benign*. The false suppression is caused by the messages forged by the adversary who impersonates the sender. This can be viewed as a binomial distribution $B(M, \frac{1}{2^l})$. The false suppression rate can thus be estimated by

$$P_{false-sup} = 1 - \sum_{i=0}^m \frac{M!}{i!(M-i)!} \left(\frac{1}{2^l}\right)^i \left(1 - \frac{1}{2^l}\right)^{M-i}$$

Figure 2 shows the false suppression rate under different settings of l , m , and M . We can clearly see that the false suppression rate is very low for a reasonable setting of l , m and M .

Overheads: Compared with the baseline approach, the additional storage overhead at sensor nodes mainly comes from the public keys of the trusted server

and the system users. The additional computation overhead at sensor nodes mainly comes from the signature verification. Since the DoS attacks against signature verification can be greatly reduced by our weak authentication method, we believe that supporting broadcast query using ECC-based signature schemes is practical for the current generation of sensor networks.

Our weak authentication scheme needs additional space for committing values at every broadcast message. Let b denote the average number of neighbor nodes for every sensor node in the network. The average overhead is about $b \times l$ bits for every broadcast message from a sensor node. We argue that this additional communication overhead is usually affordable. First, in many applications, most sensor nodes only have a small number of neighbors in their radio ranges. It is unlikely to see a very large b in practice. Second, the value b can be further reduced in many flooding methods [7–9] where every broadcast has the receiver size that is much smaller than the neighbor size. The reason is that two neighbor nodes usually share a large number of common neighbor nodes, and it is not necessary to have redundant transmission. Third, in cases where the committing values cannot fit into one packet, we can simply re-broadcast the same message multiple times with different set of committing values every time.

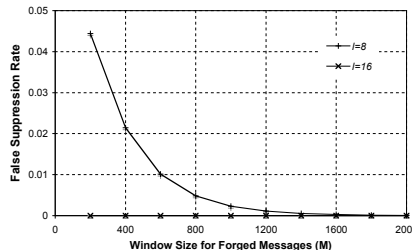


Fig. 2. False suppression rate under different settings of l and M . Assume $m = 0.01M$.

4 Simulation Evaluation

The main focus of our simulation experiments is the security as well as the performance of the broadcast query. The simulation study will focus on two aspects of the proposed weak authentication method, *its impact on the broadcast protocol* and *its performance in dealing with the DoS attacks against the signature verification*.

Many energy efficient flooding mechanisms could be used for broadcast [7–9]. However, most of them are developed for wireless ad hoc networks and do not work well in sensor networks due to the limited bandwidth and the lossy channel. Instead, we use a naive broadcast protocol in our evaluation. In this method, when a sensor node receives an authentic broadcast message for the first time, it will re-broadcast it at a probability of P_r .

In our simulation, we randomly deploy 5,093 sensor nodes in a field of size 1000×1000 square meters. Every two sensor nodes can talk to each other if they are no more than 50 meters away. Thus, there are 40 neighbor nodes on average for every node. We assume there are 40 bytes available in each packet for committing values. The broadcast will always start from the center of the field.

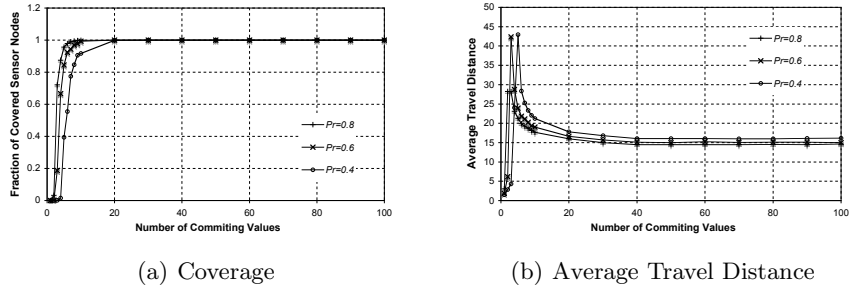


Fig. 3. Impact on broadcast protocol. Assume channel loss rate $P_l = 0.2$.

Every node will only pick b neighbors for re-broadcasting. Figure 3 shows the *coverage* (the fraction of sensor nodes that receives the broadcast message) and the *average travel distance* (the average number of hops a broadcast message travels to reach a sensor node) under different settings. The coverage affects the effectiveness of the broadcast, and the average travel distance affects the latency of the broadcast. From the figure, we can see that increasing b does improve the coverage and reduce the latency. However, after certain point (e.g., 20), increasing b will not generate much benefit. Thus, as long as b is large than certain value such as 20, we will not see big differences in terms of the performance of the broadcast. As a result, the length l of a committing value can be from 8 bits to 16 bits given 40 bytes space. The simulation result in the next part will further show that this is enough for us to effectively defend against the DoS attacks. Note that when b is very small, we will see a small average travel distance. The reason is that broadcast stops quickly after a few hops due to a small b .

We now show that the DoS attacks against signature verification can be mitigated significantly using our approach. We conducted two set of experiments. In the first set of experiments, the attacker impersonates a benign node; in the second set of experiments, the attacker launches the DoS attacks through a compromised sensor node. Clearly, in the second case, the attacker can always pass the weak authentication if he wants.

Figure 4 shows the performance of our protocol in dealing the DoS attacks on signature verification. The *suppression point* in the figure is defined as the point (i.e., the index of the fake broadcast message) where a sensor node starts to suppress the broadcast

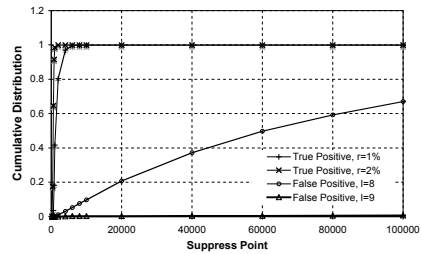


Fig. 4. Performance under DoS attacks. r is the fraction of forged messages with correct committing values (generated by a compromised node).

where a sensor node starts to suppress the broadcast

message from a given neighbor node. For example, a suppression point of 500 means that the sensor node starts to suppress the broadcast message after receiving the 500-th fake message. As discussed before, this usually happens when *the neighbor node is malicious* or *the attacker has impersonated this neighbor node for a long period of time*. From the figure, we can see that impersonating a benign node will only generate a small impact on sensor nodes, while launching attacks through a compromised node actually reveals the identity of the compromised sensor node and will be suppressed. In conclusion, our weak authentication approach effectively thwarts the DoS attacks against signature verification.

5 Related Work

This section reviews current research studies on sensor network security. To establish pairwise keys between sensor nodes, researchers have developed many key pre-distribution techniques [10–13]. μ TESLA protocol was developed to provide broadcast authentication for sensor networks [6]. DoS attacks in sensor networks have been studied in [14]. Attacks on routing protocols and counter measures were studied in [15]. Wormhole attacks have been identified as a major threat to sensor networks [16]. A typical sensor network has many supporting services such as data aggregation. These services have to be protected properly. Several techniques were developed to protect in-network processing [17, 18]. Security issues in localization and time synchronization has been studied in [19, 20]. This paper considers a critical security service, access control, in sensor networks. This is a useful service that is complementary to the above studies.

6 Conclusion and Open Problems

This paper studies the problem of access control in sensor networks. The paper develops a number of techniques to provide a practical and distributed access control method for sensor networks. There are a number of open problems. First, when there are huge number (e.g., millions) of sensor nodes and the capability string are long and detailed, the storage overhead at a user could be a big problem. Second, in the proposed technique, a malicious user can delegate its privilege to a large number of fake users and exhaust the memory at sensor nodes. It is interesting to study ideas to address these issues.

Acknowledgment The authors would like to thank the anonymous reviewers for their valuable comments.

References

1. Crossbow Technology Inc.: Wireless sensor networks. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm Accessed in February 2006.
2. Hartung, C., Balasalle, J., Han, R.: Node compromise in sensor networks: The need for secure systems. Technical Report CU-CS-990-05, U. Colorado at Boulder (Jan. 2005)

3. Zhang, W., Song, H., Zhu, S., Cao, G.: Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks. In: Proceedings of ACM Mobihoc'05. (2005)
4. Wang, H., Li, Q.: Distributed user access control in sensor networks. In: The International Conference on Distributed Computing in Sensor Systems (DCOSS '06). (2006)
5. Gura, N., Patel, A., Wander, A.: Comparing elliptic curve cryptography and rsa on 8-bit CPUs. In: Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004). (August 2004)
6. Perrig, A., Szewczyk, R., Wen, V., Culler, D., Tygar, D.: SPINS: Security protocols for sensor networks. In: Proceedings of Seventh Annual International Conference on Mobile Computing and Networks. (July 2001)
7. Lim, H., Kim, C.: Multicast tree construction and flooding in wireless ad hoc networks. In: Proceedings of ACM Modeling, Analysis, and Simulation of Wireless and Mobile Systems. (2000)
8. Peng, W., Lu, X.: On the reduction of broadcast redundancy in mobile ad hoc networks. In: Proceedings of ACM International Symposium on Mobile and Ad Hoc Networking and Computing. (2000)
9. Wu, J., Dai, F.: Broadcasting in ad hoc networks based on self-pruning. In: Proceedings of INFOCOM. (2003)
10. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security. (November 2002) 41–47
11. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Research in Security and Privacy. (2003) 197–213
12. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03). (October 2003) 52–61
13. Du, W., Deng, J., Han, Y.S., Varshney, P.: A pairwise key pre-distribution scheme for wireless sensor networks. In: Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03). (October 2003) 42–51
14. Wood, A.D., Stankovic, J.A.: Denial of service in sensor networks. *IEEE Computer* **35**(10) (2002) 54–62
15. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: Attacks and countermeasures. In: Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications. (May 2003)
16. Hu, Y., Perrig, A., Johnson, D.: Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In: Proceedings of INFOCOM 2003. (April 2003)
17. Du, W., Deng, J., Han, Y.S., Varshney, P.K.: A witness-based approach for data fusion assurance in wireless sensor networks. In: Proceedings of IEEE Global Communications Conference (GLOBECOM 03). (December 2003)
18. Przydatek, B., Song, D., Perrig, A.: SIA: Secure information aggregation in sensor networks. In: Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys '03). (Nov 2003)
19. Liu, D., Ning, P., Du, W.: Attack-resistant location estimation in wireless sensor networks. In: Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN '05). (April 2005)
20. Sun, K., Ning, P., Wang, C.: Fault-tolerant cluster-wise clock synchronization for wireless sensor networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)* **2**(1) (September 2005) 177–189