Available online at www.sciencedirect.com

ScienceDirect

Ad Hoc Networks 5 (2007) 3–13

Ad Hoc Networks

www.elsevier.com/locate/adhoc

Access control in wireless sensor networks ☆

Yun Zhou, Yanchao Zhang, Yuguang Fang *

Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, United States

Available online 5 July 2006

Abstract

Nodes in a sensor network may be lost due to power exhaustion or malicious attacks. To extend the lifetime of the sensor network, new node deployment is necessary. In military scenarios, adversaries may directly deploy malicious nodes or manipulate existing nodes to introduce malicious "new" nodes through many kinds of attacks. To prevent malicious nodes from joining the sensor network, access control is required in the design of sensor network protocols. In this paper, we propose an access control protocol based on *Elliptic Curve Cryptography* (ECC) for sensor networks. Our access control protocol accomplishes node authentication and key establishment for new nodes. Different from conventional authentication methods based on the node identity, our access control protocol includes both the node identity and the node bootstrapping time into the authentication procedure. Hence our access control protocol cannot only identify the identity of each node but also differentiate between old nodes and new nodes. In addition, each new node can establish shared keys with its neighbors during the node authentication procedure. Compared with conventional sensor network security solutions, our access control protocol can defend against most well-recognized attacks in sensor networks, and achieve better computation and communication performance due to the more efficient algorithms based on ECC than those based on RSA.

© 2006 Elsevier B.V. All rights reserved.

Keywords: Sensor networks; Access control; RSA; Diffie–Hellman; ECC

1. Introduction

Due to the significant advances of hardware manufacturing technology and developments of efficient software algorithms, a network of a large number of small and low-cost sensors through wireless communications, i.e., *Wireless Sensor Network*, has become technically and economically feasible and been drawing intensive interests from both academic and industrial areas [1]. Usually, a wireless sensor network is deployed in a designated area without any fixed infrastructure where sensor nodes cooperate with each other to perform various applications.

To save manufacturing cost, a sensor node is usually built as a small device, which has limited memory, a low-end processor, and is powered by a battery [2]. The constrained resources result in limited computation and communication capabilities.

☆ This work was supported in part by the US Office of Naval Research under grant N000140210464 (Young Investigator Award), and by the US National Science Foundation under grant ANI-0093241 (CAREER Award) and under grant ANI-0220287.

* Corresponding author. Tel.: +1 352 846 3043; fax: +1 352 392 0044.

E-mail addresses: yzufl@ufl.edu (Y. Zhou), yczhang@ufl.edu (Y. Zhang), fang@ece.ufl.edu (Y. Fang).

After several weeks or months of operation, some nodes in the network may exhaust their power because of the uneven distribution of traffic load. Therefore new node deployment is necessary in this case.

Besides the natural loss of sensor nodes, a sensor network is also susceptible to malicious attacks in unattended and hostile environments. Some sensor nodes may be destroyed by adversaries so that the entire network may become useless. Hence, new sensor nodes need to be deployed. However, an adversary can also deploy malicious nodes into the network. These malicious nodes may easily eavesdrop messages or insert false reports [2]. In addition, an intelligent adversary may launch tricky attacks from the inside of the sensor network by manipulating existing sensor nodes. A sensor node may be compromised due to the lack of tamper resistance [3] so that all the secrets in it are exposed to the adversary. Then the adversary may use the compromised node to launch other more serious attacks [4–8], which cause fatal havoc [9,10] in the sensor network.

Recently, many schemes [2,11–16] were proposed to protect sensor networks. They may prevent external attackers from eavesdropping messages or inserting false reports. However, they can hardly defend against internal attacks [4–8]. Though several techniques [4–8] were proposed to counteract the internal attacks, each of them is only targeted to one specific attack by using different approaches and hardware assumptions. It is very difficult to integrate those techniques into a uniform hardware platform. Even if the integration is possible, it may cost a lot of resources and deviate from the low cost consideration.

In this paper, we analyze the internal attacks in [4–8]. We observe that the common trick under these attacks is that they manipulate existing nodes to introduce malicious "new" nodes, which are indistinguishable from legitimate new nodes under current sensor network security technology. Those introduced "new" nodes could be accepted by other normal nodes as legitimate ones. Based on this observation, we design an access control protocol for sensor networks to prevent malicious nodes, no matter whether they are directly deployed by adversaries or introduced "new" ones, from participating in sensor networks. A new node should prove that it not only has correct identity but also is truly new to be admitted into the sensor network. Besides the node identity which is widely used in authentication, we introduce the node bootstrapping time, which is the time when the new node bootstraps itself to join the sensor network, into the authentication procedure to differentiate malicious "new" nodes, which are actually old nodes, from legitimate new nodes. Unlike the conventional approaches in [4–8] that attempt to detect malicious nodes after they join sensor networks, our access control protocol can prevent malicious nodes from joining sensor networks at the very beginning. Moreover, key establishment is also included in our access control protocol to help the new node establish shared keys with its neighbors so that it can perform secure communications with them.

The rest of this paper is organized as follows. We analyze most typical internal attacks in Section 2. The details of our access control protocol are described in Section 3. Some security analysis and performance evaluations are carried out in Sections 4 and 5. We finally conclude the paper in Section 6.

2. Review of attacks

In military applications sensor networks are often deployed in hostile environments. An adversary can directly *deploy malicious nodes*, say node B in Fig. 1(a), to eavesdrop messages sent out or received by normal nodes or even inject false reports to disrupt the network functionalities [2,9,10]. In the *Sybil* attack [4,17], a malicious node illegitimately takes on multiple identities. The impersonated identities may belong to existing nodes or non-existing nodes. The malicious node may be deployed directly by adversaries or just a compromised one. In Fig. 1(b), for example, a node B is compromised and then impersonates other identities, e.g., node C. From the point of the view of node A, it is just like a new node C coming out in its vicinity. It has been shown that the Sybil attack may pose a serious threat to distributed storage, routing protocols [10], data aggregation, voting, fair resource allocation, misbehavior detection [4], etc. In the *node replication* attack [5], an adversary intentionally puts many replicas of a compromised node, say node B in Fig. 1(c), at many places, say the vicinity of node A, in the network to incur inconsistency. Node A may take node B as its new neighbor. Like the Sybil attack, the node replication attack can also render adversaries the abilities to subvert data aggregation, misbehavior detection and voting protocols by injecting false data or suppressing legitimate data [5]. In the *Wormhole* attack [6], an

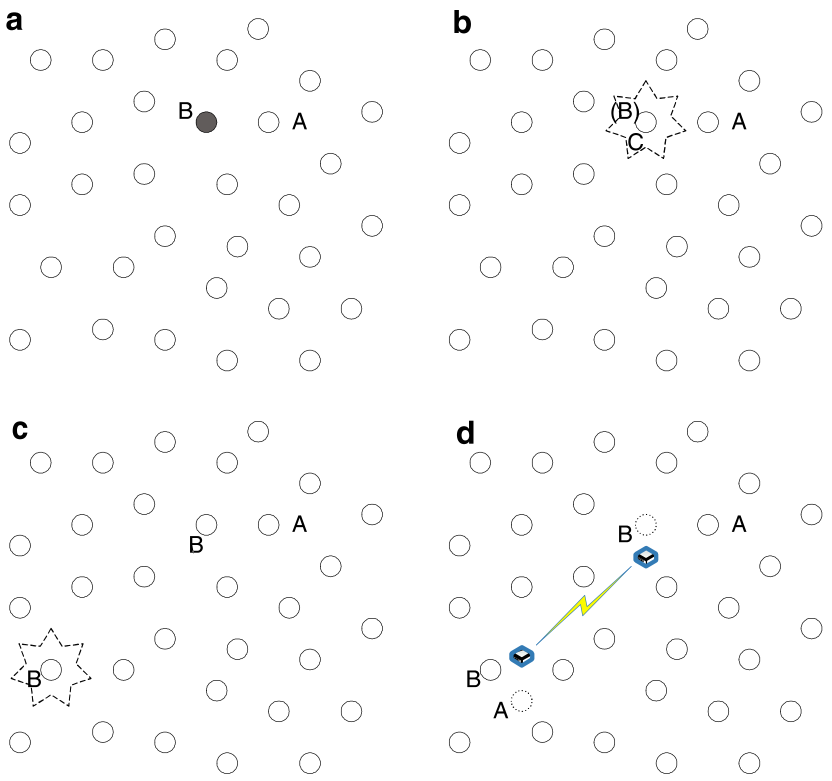


Fig. 1. Attacks. (a) A malicious node *B* is deployed in the vicinity of node *A*. (b) The Sybil attack in which a compromised node *B* claims a new identity *C* in the vicinity of node *A*. (c) The node replication attack in which a copy of the compromised node *B* is deployed in the vicinity of node *A*. (d) The Wormhole attack in which an adversary tunnels packets between node *A* and node *B* so that one node finds a new neighbor which is actually the image of the other node.

adversary tunnels packets between two distant places in the sensor network. In Fig. 1(d), for example, the adversary deploys two special devices into the vicinities of node *A* and node *B*, respectively. These two devices share a secret broadband channel and tunnel packets through the secret channel between nodes *A* and *B*. The consequence is that node *A* may find a new node *B* coming out in its neighborhood, and vice versa. This attack may distort the network topology by making two distant nodes believe they are neighbors, thus becoming a serious attack to routing protocols [6].

3. Our protocol

3.1. Outline

New node deployment is inevitable because of the loss of sensor nodes. A deployed new node, however, may not be a legitimate one as is shown in Section 2. It may be a malicious node directly deployed by adversaries, or an introduced "new" node. Those malicious "new" nodes are indistinguishable from legitimate new nodes under current sensor network security technology, and thus will be accepted by other normal nodes as legitimate ones. To prevent malicious nodes from joining sensor networks, access control should be enforced to control sensor node deployment. Usually, an access control mechanism should accomplish two tasks:

1. *Node authentication*: Through authentication a deployed node proves its identity (ID) to its neighboring nodes and proves that it has the right to access the sensor network.
2. *Key establishment*: Shared keys should be established between a deployed node and its neighboring nodes to protect communications.

A preloaded public key certificate can be used to prove the identity of a new node. When the new node is deployed into the sensor network, its neighbors may verify the certificate to check whether the new node has a legitimate identity. By using this ID authentication, adversaries are prevented from directly deploying malicious nodes because they do not have corresponding certificates. However, the ID authentication is not enough to protect the sensor network. In the Sybil attack, the node replication attack and the Wormhole attack, an adversary in fact could manipulate existing nodes to introduce malicious "new" nodes. Those old nodes have preloaded certificates, so the "new" nodes also have legitimate identities. Hence, we need to differentiate those old nodes from new nodes to further protect the sensor network.

A solution to solve the problem is to involve a timestamp into the authentication procedure. It is a common solution to solving the freshness problems in our real lives. For example, the tickets we buy for movies or football games carry timestamps which show when the tickets are valid. The similar idea can also be applied to the design of our access control protocol for sensor networks. After a sensor node is deployed into a sensor network, it will bootstrap itself at a preset time to join the sensor network. The difference between an old node and a new node is that they have different bootstrapping times. Hence, we may use the *bootstrapping time* as the timestamp into our access control protocol.

Our access control protocol uses a preloaded certificate which includes both ID information and bootstrapping time to authenticate the identity of a new node. The certificate is generated by a *Certification Authority* (CA), e.g., the administrator of the sensor network. In the certificate the node ID information and its bootstrapping time are signed by CA's private key to protect their integrities, so that adversaries cannot falsify the ID and the bootstrapping time. When the new node is deployed into the sensor network, it can show its certificate to its neighbors. The neighbors can verify the ID and the bootstrapping time with the CA's public key. A new node can be accepted into the sensor network only if it has a correct identity and its bootstrapping time is within a tolerance period of current time.

Our cryptographic tool is the *Elliptic Curve Cryptography* (ECC) [18,19]. Compared to RSA [20], ECC is seen to be the standard for the next generation cryptographic technology. The reason is that ECC can achieve the same level of security with smaller key sizes. It has been shown that 160-bit ECC provides comparable security to 1024-bit RSA and 224-bit ECC provides comparable security to 2048-bit RSA [21]. Under the same security level, smaller key sizes of ECC offer merits of faster computational efficiency, as well as memory, energy and bandwidth savings, thus ECC is better suited for the resource constrained devices. Due to the merits of ECC, our access control protocol uses 160-bit ECC as the underlying cryptographic infrastructure. Particularly, the signature operation in our protocol is based on the *Elliptic Curve Digital Signature Algorithm* (ECDSA) [21], and the shared key is established according to the Diffie–Hellman [22] algorithm over ECDLP.

3.2. Assumptions

We assume that sensor nodes are stationary so that if a node finds a new node in its neighborhood, the new node must be either a newly deployed node or a node introduced by adversaries. All sensor nodes have the same transmission range and communicate with each other via bi-directional wireless links. Each node has a unique, integer-valued, nonzero ID. We assume that all sensor nodes are loosely synchronized. Each sensor node has a preset bootstrapping time. After being deployed into the sensor network, each sensor node bootstraps itself at its bootstrapping time to join the sensor network. Two sensor nodes may have the same bootstrapping time if they are deployed simultaneously. A possible collision at the MAC layer may occur if the two nodes bootstrap themselves simultaneously. However, we assume that the MAC-layer protocol has collision resolution mechanisms to solve the problem [23]. Hence, each node can finish bootstrapping within a tolerance time interval after its bootstrapping time. Though node compromise is usually unavoidable, compromising is not a trivial job. We assume that each sensor node can sustain a tolerance time interval before it is compromised, which is also assumed by previous work [3,24].

3.3. Predeployment phase

Before a sensor network is deployed, the CA chooses a set of *network parameters* including: a finite field $\mathbb{F}_q$, where q is a large odd prime of at least 160 bits; an elliptic curve E over $\mathbb{F}_q$ (denoted by $E(\mathbb{F}_q)$ hereafter); a cyclic group $\mathbb{G} = \langle G \rangle$ of points over the elliptic curve $E(\mathbb{F}_q)$, where G is the generator of the group and has an order n of at least 160 bits, with $n > 4\sqrt{q}$; the CA's private key $\kappa \in \mathbb{Z}_n^* = \{1, 2, \ldots, n-1\}$; the CA's public key $Q = \kappa G \in \mathbb{G}$. The CA never shares its private key with anyone else. Since ECDLP is a hard problem [18,19], no one can derive the CA's private key κ from the pair $\langle G, Q\rangle$. In addition, the CA does not get involved in the network operation, so adversaries have no opportunity to directly attack the CA to get κ.

For each sensor node, say N_i, the CA preloads it with a set of *node parameters* including: the elliptic curve $E(\mathbb{F}_q)$; the cyclic group $\mathbb{G}$ over $E(\mathbb{F}_q)$; the CA's public key Q; the bootstrapping time T_i when node N_i bootstraps itself to join the sensor network; the length of bootstrapping phase L_i during which the node is allowed to join the sensor network; N_i's private key $s_i \in \mathbb{Z}_n^*$; N_i's public key $P_i = s_iG = (x_{pi}, y_{pi}) \in \mathbb{G}$, where $x_{pi}, y_{pi} \in \mathbb{F}_q$; the signature $\langle C_i, c_i\rangle$ for node N_i, where $C_i \in \mathbb{G}$ and $c_i \in \mathbb{Z}_n^*$; a hash function $H : \{0,1\}^* \to \mathbb{Z}_n^*$, which translates a binary sequence into an integer in $\mathbb{Z}_n^*$.

The signature is calculated according to ECDSA. The CA first chooses a random number $k_i \in \mathbb{Z}_n^*$ and then calculates

$$C_i = k_iG = (x_{ci}, y_{ci}), \tag{1}$$

$$c_i = k_i^{-1}(H(N_i \| T_i \| L_i \| P_i) + \kappa x_{ci}) \pmod{n}, \tag{2}$$

where "$\|$" is the concatenation operator.

3.4. Node deployment

At the very beginning, a network of sensor nodes, say hundreds or thousands of nodes, is deployed in a designated area. At a preset time, these sensor nodes bootstrap themselves and then start to establish communications. During the network operation phase, if some sensor nodes are lost, new sensor nodes need to be deployed. These new sensor nodes all have a preset bootstrapping time different from that of the previously deployed nodes. Hence, without loss of generality, we assume that sensor nodes are deployed in groups, where sensor nodes in one group have the same bootstrapping time and the length of bootstrapping phase but these values for different groups may be different.

3.5. Node authentication

Every new node should broadcast a message to inform its neighbors of its existence. For example, a new node N_i bootstraps itself at time T_i and broadcasts a message:

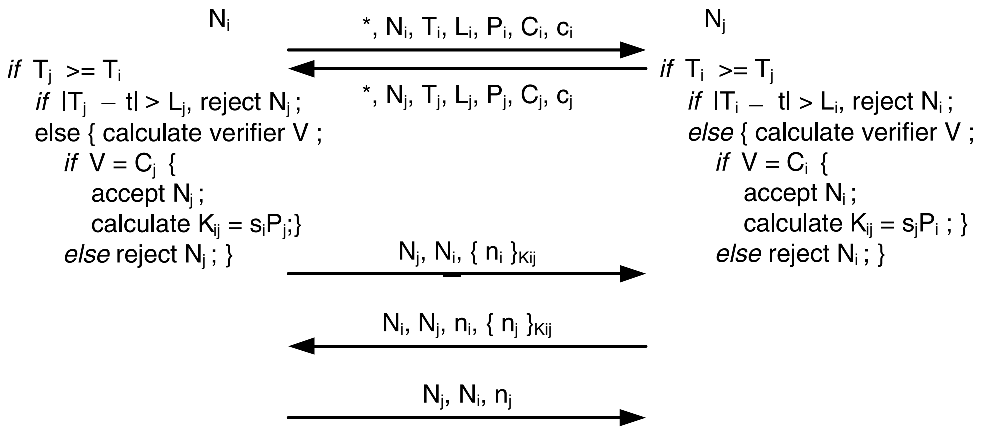
$$N_i \to * : \langle *, N_i, T_i, L_i, P_i, C_i, c_i\rangle. \tag{3}$$

Then handshakes between the new node and its neighbors can be performed for authentication. Because the neighbors of the new node may include both new nodes and old nodes, the handshakes can be divided into two cases: the handshake between new nodes (Fig. 2) and the handshake between a new node and an old node (Fig. 3).

3.5.1. Handshake between new nodes

If node N_i also hears a broadcasted message from another new node N_j, it verifies whether N_j is a legitimate new node by doing the following.

Node N_i first compares N_j's bootstrapping time T_j with its own bootstrapping time T_i. If $T_j \geqslant T_i$, then node T_j might be a new node. Actually $T_j = T_i$



N_i
N_j
*, N_i, T_i, L_i, P_i, C_i, c_i
*, N_j, T_j, L_j, P_j, C_j, c_j
if T_j >= T_i
if |T_j – t| > L_j, reject N_j ;
else { calculate verifier V ;
if V = C_j {
accept N_j ;
calculate K_{ij} = s_iP_j;}
else reject N_j ; }
if T_i >= T_j
if |T_i – t| > L_i, reject N_i ;
else { calculate verifier V ;
if V = C_i {
accept N_i ;
calculate K_{ij} = s_jP_i ; }
else reject N_i ; }
N_j, N_i, { n_i }$_{Kij}$
N_i, N_j, n_i, { n_j }$_{Kij}$
N_j, N_i, n_j

Fig. 2. Handshake between two new nodes. Each of them checks the validity of the bootstrapping time (in this case $T_j = T_i$ because N_i and N_j are both new nodes) and the identity of the other node by ECDSA. The shared key between them is verified through a challenge–response procedure.

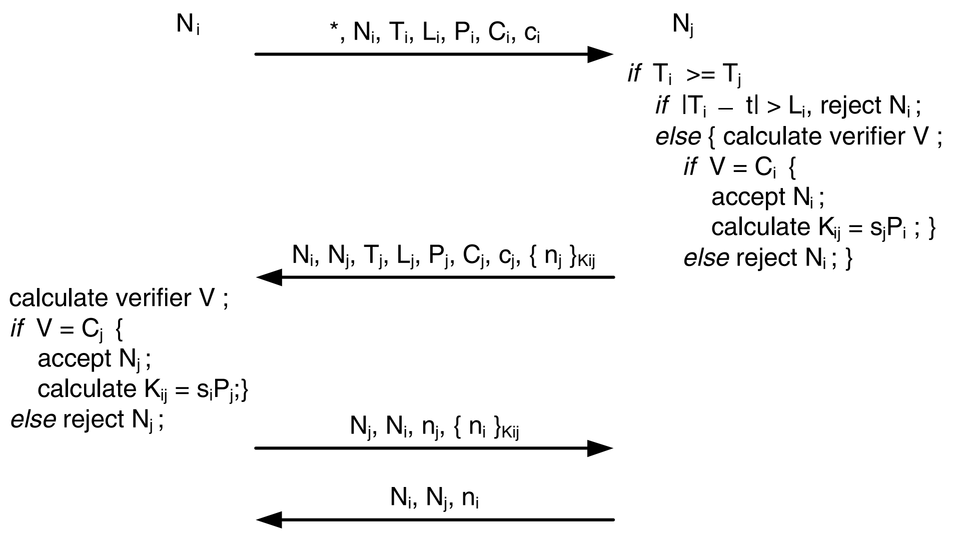


Fig. 3. Handshake between a new node and an old node. The old node checks the validity of the bootstrapping time and the identity of the other node by ECDSA. The new node just checks the identity of the old node by ECDSA. The shared key between them is verified through a challenge–response procedure.

if N_i and N_j are both new nodes. The reason of using "$\geqslant$" here is to maintain the software compatibility so that this procedure can also be used by an old node to authenticate a new node (refer to Fig. 3). Node N_i proceeds to verify whether node N_j is a new node by comparing T_j with its current time t. If T_j is out of date ($|T_j - t| > L_j$), node N_i simply drops the received message. Otherwise, node N_i continues to verify N_j's identity by ECDSA. Specifically, node N_i computes

$$u_1 = H(N_j \| T_j \| L_j \| P_j), \quad (4)$$

$$u_2 = c_j^{-1} u_1 \pmod{n}, \quad (5)$$

$$u_3 = c_j^{-1} x_{cj} \pmod{n}, \quad (6)$$

$$V = u_2 G + u_3 Q. \quad (7)$$

If $V = C_j$, node N_i can make sure that node N_j is a legitimate new node. This is because if the signature is valid, the verification equation holds:

$$\begin{aligned} V &= u_2 G + u_3 Q = c_j^{-1} u_1 G + c_j^{-1} x_{ci} Q \\ &= c_j^{-1}(H(N_j \| T_j \| L_j \| P_j) + \kappa x_{ci}) G = k_j G = C_j. \end{aligned} \quad (8)$$

After node N_i verifies the identity of node N_j, it calculates a shared key with its private key and N_j's public key, i.e., $K_{ij} = s_i P_j = s_i s_j G$. Following the same procedure, node N_j can verify the identity of node N_i after it hears the broadcasted message from node N_i and calculate a shared key as $K_{ij} = s_j P_i = s_i s_j G$.

Node N_i and node N_j can make sure that each other does have the shared key by following the *challenge–response* procedure. Node N_i just selects a nonce n_i, encrypts it and sends it to node N_j. If node N_j has the shared key, it can decrypt the nonce n_i. Then node N_j sends back a message including the nonce n_i and an encrypted nonce n_j chosen by itself to node N_i. Node N_i can also decrypt the nonce n_j and return it to node N_j. The handshake between node N_i and node N_j is depicted in Fig. 2.

3.5.2. Handshake between a new node and an old node

When an old node N_j hears the broadcasted message from the new node N_i, it also checks the validity of N_i's bootstrapping time and then verifies N_i's identity (Fig. 3). After that, node N_j calculates a shared key with its private key and N_i's public key, selects a nonce n_j, encrypts the nonce with the shared key, and replies with the message:

$$N_j \rightarrow N_i : \langle N_i, N_j, T_j, L_j, P_j, C_j, c_j, \{n_j\}_{K_{ij}} \rangle. \quad (9)$$

Node N_i does not need to check the validity of N_j's bootstrapping time because N_j is not a new node. Adversaries may attack our access control protocol by utilizing this point. We will analyze this in Section 4. Node N_i simply verifies N_j's identity by following ECDSA. Then node N_i can decrypt the nonce n_j and return it to N_j to show that it is a legitimate new node. Node N_i also challenges node

N_j by sending an encrypted nonce n_i and requiring N_j to return it. The whole handshake is depicted in Fig. 3.

3.6. Key establishment

During the node authentication procedure, the new node N_i has already established shared keys with its neighbors, e.g., N_j. Because no efficient algorithm can solve ECDLP within less than exponential time, we can expect that adversaries cannot calculate the private keys s_i and s_j given pairs $\langle G, s_iG\rangle$ and $\langle G, s_jG\rangle$. Hence, the shared key K_{ij} is kept secret even if adversaries eavesdrop transmitted public keys. Later, the shared key K_{ij} between node N_i and node N_j can be used to derive different keys for multiple security services, such as message encryption and message authentication [2].

4. Security analysis

4.1. New node deployment

By authentication, our access control protocol can prevent adversaries from directly deploying malicious nodes into sensor networks. Because adversaries do not know the private key of the CA, he/she cannot falsify certificates for malicious nodes. Our access control protocol can effectively defend against the Sybil attack, the node replication attack, and the Wormhole attack. By including the bootstrapping time in our access control protocol, a new node is only allowed to join the sensor network during its bootstrapping phase. After that it becomes an old node. Hence, malicious "new" nodes (Section 2) are prevented from joining the sensor network at the very beginning, because they do not have the proper bootstrapping time, and they are prevented from falsifying the latest bootstrapping time which does not match their certificates.

4.2. Eavesdropping and false reports injection

When a new node passes the authentication procedure, it has already established shared keys with its neighbors by following the Diffie–Hellman algorithm over ECDLP. The shared keys can be used to secure communications among sensor nodes. Hence, external adversaries are prevented from eavesdropping or injecting false reports into the sensor network.

4.3. Node compromise

Usually node compromise cannot be prevented in sensor networks, unless future advances of hardware design and manufacturing could provide stronger tamper resistance [3]. Our access control cannot eliminate the node compromise problem, but it can prevent adversaries from spreading the impact of node compromise across the entire network. Two direct results of node compromise, the Sybil attack and the node replication attack, can be prevented by our access control protocol after the node bootstrapping phase. Moreover, based on the ECC public key infrastructure, each sensor node does not know the private keys of other nodes, and each shared key is only known to two neighboring nodes who established it. Even if an adversary compromises a node, he/she can only know what the compromised node knows, but not the shared keys between other non-compromised nodes. Hence, the impact of node compromise is limited to the vicinity of the compromised node.

If an adversary could compromise a sensor node during its bootstrapping phase, he/she might use it to launch other attacks. However, node compromising is not a trivial task. Usually a sensor node is designed to be able to sustain compromise for a certain time interval [3]. The node bootstrapping phase, however, is usually very short, and in practice it is reasonable to expect it to be shorter than the time needed to compromise the node [24]. Hence we do not need to worry about node compromise during the node bootstrapping phase.

4.4. Attacks to access control

Our access control protocol tries to solve the new node deployment problem in hostile environments. During the handshake between a new node and an old node, the bootstrapping time of the new node is verified by the old node, but the new node does not check the bootstrapping time of the old node because the old node has been involved in the sensor network. An adversary may take this opportunity to trick the new node into establishing communications with malicious old nodes.

One scenario is that an adversary might introduce a malicious node through the Sybil attack or the node replication attack into the area where the new node is to be deployed. When the new node is deployed, it might establish communications with the malicious node. To make the attack successful, however, the adversary has to activate the malicious node at the same time when the new node bootstraps itself and expects that no other old nodes exist in that area. Otherwise, the introduced malicious node can be detected by other old nodes because the malicious node is heard by those old nodes as a new node but it does not have the correct bootstrapping time. Under this strict condition, the probability of this attack is rather small.

A similar scenario is that an adversary might launch the Wormhole attack to establish a tunnel between a new node and another distant old node so that these two nodes might establish communications through handshakes. To make the attack successful, the adversary still has to establish the tunnel at the same time when the new node bootstraps itself and expects that no other old nodes exist around the new node. Otherwise, the old nodes around the new node can detect the Wormhole because they can find a "new" node in their neighborhoods, which is actually an image of the old node at the other end of the Wormhole. We can expect that the probability of this attack is also very small under the strict condition.

Another case is that the adversary just compromises an old node without doing any tricks to spread its impact. The compromised node stays at its original location and follows the normal network protocols. If the new node is deployed into the vicinity of the compromised node, they could establish communications. This attack is just the node compromise attack and currently no solutions can solve the problem. Our access control protocol cannot prevent this attack, either, but the impact of the attack is limited to the vicinity of the compromised node.

5. Evaluation

5.1. ECC vs. RSA

The length of the bootstrapping phase is critical for the security performance of our access control protocol. The shorter the bootstrapping phase is, the less opportunities adversaries have to attack the sensor network. Hence a short bootstrapping phase is desirable to keep the sensor network safe.

Our protocol uses ECC rather than RSA and Diffie–Hellman over DLP because ECC is more efficient for the same security level. In our protocol, the most expensive operation is the *point multiplication* of the form kP for $k \in \mathbb{Z}_n^*$ and $P \in \mathbb{G}$. Every sensor node needs to perform only three point multiplications over an elliptic curve: two for node authentication and one for key establishment. TinyPK [16] uses RSA to authenticate external parties and Diffie–Hellman over DLP to establish shared keys between external parties and sensor nodes. It requires three *modular exponentiation* operations over integer rings for each sensor node: one RSA public key operation and one RSA private key operation for node authentication and one DLP operation for key establishment. It has been shown in [16,25] that a point multiplication needs less computation time than a modular exponentiation unless the exponent is chosen as some specific value. In TinyPK [16], a public exponent $e = 3$ is chosen for computational simplicity, and a 1024-bit RSA modular exponentiation with $e = 3$ on MICA1 Motes [26] needs 14.5 s. The DLP of 2^x is evaluated in [16,25]. It shows that a 1024-bit modular exponentiation 2^x, where x is at least 160 bits, needs more than 50 s on both MICA1 and MICA2 Motes [26]. However, a 163-bit point multiplication of ECC on MICA2 Motes requires only 34s [25]. If assembly languages are used in implementation, much more decrease of computing time can be achieved. Gura et al. [27] evaluated the assembly language implementations of ECC and RSA on the Atmel ATmega128 processor [28], which is popular for sensor platform such as Crossbow MICA Motes. In their implementation, a 160-bit point multiplication of ECC requires only 0.81 s, while 1024-bit RSA public key operation and private key operation require 0.43 s and 10.99 s, respectively. Obviously, ECC is more computational efficient, especially for assembly language implementations, which makes ECC realistic on current sensor hardware platforms. This means every sensor node can finish bootstrapping in a very short time interval. With the fast advance of hardware technology, we believe the bootstrapping phase can be further reduced in future.

In wireless sensor networks, the transmission energy consumption rate could be over three orders of magnitude greater than the energy consumption rates for computing [29]. Most of the performance overhead is attributable to the increase in packet size [30]. Compared with a 1024-bit RSA signature, our access control protocol only introduces a 480-bit signature when 160-bit ECC is used. Hence by using ECC instead of RSA our protocol can achieve much more energy and bandwidth savings.

5.2. Comparison with related work

Because currently no solutions can prevent node compromise in sensor network, the best we can do is to limit the impact of node compromise to the vicinity of the compromised nodes, i.e., prevent adversaries from launching network-scale attacks based on compromised nodes. Most of symmetric key techniques, including randomly predistributed keys [12,13], ID-based keys [14,15], and location-based keys [31–33] try to improve the resilience to node compromise by increasing the least number of sensor nodes that an adversary needs to compromise to destroy the entire network security architecture. These schemes can tolerate a certain number of compromised nodes. TinyPK [16] is more resilient to node compromise because of the use of RSA. It may prevent adversaries from spreading the impact of node compromise by launching the Sybil attack, but it cannot detect the node replication attack because the copies of the compromised nodes also have legitimate certificates.

To defend against the Sybil attack, several potential methods were proposed in [4]. One method is the radio resource testing, which assumes that each node has enough radio resources and requires several rounds of broadcasting over multiple channels, thus leading to a large communication overhead. Another method is to use the ID-based symmetric keys. Particularly, each sensor node is preloaded with a set of keys which are selected from a global key pool by its node ID. The ID of a suspect node is challenged by a set of validating nodes based on the keys shared between the suspect node and the validating nodes. Besides the large amount of communication overhead, this method may fail if many sensor nodes are compromised so that most of the keys in the global key pool are exposed.

Conventional methods to defend against the node replication attack [5] usually include centralized computing based on node locations or the number of simultaneous connections, which is vulnerable to the single-point failure. Distributed detection of the node replication attack was proposed in [5], where each node is assumed to know its location and it is required to send its location to a set of witness nodes. If a witness node finds a contradiction in the location claims of a suspect node, this suspect node must be a replicated one. Obviously, this method may introduce a lot of communication overhead. The authentication procedure in our protocol is performed locally, thus avoiding much more communication overhead. Moreover, our protocol does not require each node to know its own location.

To defend against the Wormhole attack, Hu et al. proposed to use *packet leashes* [6] to limit the maximum range over which packets can be tunneled. They require that each node either know its location or have a tightly synchronized clock so that this information can be used to calculate the maximum distance that a relayed packet could travel. *Directional antennas* [7] were also used to defend against the Wormhole attack. However, these defenses are targeted to ad hoc networks and require expensive hardware devices, which may be infeasible for most resource constrained sensor networks. Our protocol does not require location information and only needs loose synchronized clock. Wang and Bhargava [8] proposed to use centralized computing to defend against the Wormhole attack in sensor networks, in which a controller collects all nodes' location information to reconstruct the network topology such that any topological distortion may be visualized. This approach, however, causes much intensive communication overhead and is only suitable for static Wormhole. If adversaries move around in the entire network, the location of the Wormhole will change dynamically. Each location change will trigger a new round of execution of the topology reconstruction algorithm. Our protocol can prevent dynamic Wormhole by only involving localized authentication, thus can save a lot of communication overhead.

6. Conclusion

Currently little work has been reported to address the access control problem in sensor networks. Though many proposals [2,11–16,25,31–33] try to secure sensor networks, adversaries can still attack the networks [4–8] by manipulating old nodes to introduce malicious "new" nodes. In this paper, we design an access control protocol to prevent malicious nodes, which may be directly deployed or just old nodes manipulated by adversaries, from participating in sensor networks. Besides the node identity authentication, we introduce the node bootstrapping time into the node authentication procedure to differentiate malicious nodes from legitimate new nodes. Unlike the conventional approaches in [4–8] that try to detect malicious nodes after they join sensor networks, our access control protocol can prevent malicious nodes from joining sensor networks at the very beginning. In addition, key establishment is also realized in our access control protocol to help the new node establish shared keys with its neighbors so that it can perform secure communications with them. Compared with the conventional sensor network security solutions, our access control protocol can defend against most of the notorious attacks in sensor networks, and achieve better computation and communication performance due to the usage of the more efficient algorithms based on Elliptic Curve Cryptography than those based on RSA.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, IEEE Communication Magazine 40 (8) (2002) 102–114.

[2] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, D.E. Culler, SPINS: Security protocols for sensor networks, Wireless Networks 8 (September) (2002) 521–534.

[3] Ross Anderson, Markus Kuhn, Tamper resistance – a cautionary note, in: Proceedings of the 2nd USENIX Workshop on Electronic Commerce, Oakland, California, 18–21 November 1996, pp. 1–11.

[4] J. Newsome, E. Shi, D. Song, A. Perrig, The sybil attack in sensor networks: analysis & defenses, in: The 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04), Berkeley, California, USA, 26–27 April 2004.

[5] B. Parno, A. Perrig, V. Gligor, Distributed detection of node replication attacks in sensor networks, in: IEEE S&P'05, 2005.

[6] Y. Hu, A. Perrig, D.B. Johnson, Pachet leashes: a defense against wormhole attacks in wireless networks, in: IEEE INFOCOM'03, 2003.

[7] L. Hu, D. Evans, Using directional antennas to prevent wormhole attacks, in: The 11th Annual Network and Distributed System Security Symposium (NDSS'04), San Diego, California, 5–6 February 2004.

[8] W. Wang, B. Bhargava, Visualization of wormholes in sensor networks, in: Proceedings of the 2004 ACM Workshop on Wireless Security (Wise'04), Philadelphia, PA, USA, 1 October 2004.

[9] A. Wood, J. Stankovic, Denial of service in sensor networks, IEEE Computer (October) (2002) 54–62.

[10] Chris Karlof, David Wagner, Secure routing in wireless sensor networks: attacks and countermeasures, in: First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03), May 2003.

[11] C. Karlof, N. Sastry, D. Wagner, TinySec: a link layer security architecture for wireless sensor networks, in: The Second ACM Conference on Embedded Networked Sensor Systems (SensSys'04), Baltimore, Maryland, November 2004.

[12] L. Eschenauer, V. Gligor, A key management scheme for distributed sensor networks, in: The Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02), Washington DC, 2002.

[13] Haowen Chan, Adrian Perrig, Dawn Song, Random key predistribution schemes for sensor networks, in: Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P'03), 11–14 May 2003, p. 197.

[14] W. Du, J. Deng, Y.S. Han, P.K.Varshney, A pairwise key pre-distribution scheme for wireless sensor networks, in: The Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03), Washington, DC, 27–30 October 2003.

[15] D. Liu, P. Ning, Establishing pairwise keys in distributed sensor networks, in: The Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03), Washington, DC, 2003.

[16] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, P. Kruus, TinyPK: securing sensor networks with public key technology, in: Proceedings of the 2nd ACM Workshop on Security of Ad hoc and Sensor Networks (SASN'04), Washington, DC, USA, 25 October 2004.

[17] J.R. Douceur, "The Sybil attack," First International Workshop on Peer-to-Peer Systems (IPTPS'02), March 2002.

[18] N. Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation 48 (1987) 203–209.

[19] V. Miller, Uses of elliptic curves in cryptography, in: Advances in Cryptology – CRYPTO'85, Lecture Notes in Computer Science, vol. 218, Springer-Verlag, Berlin, 1986, pp. 417–426.

[20] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (2) (1978) 120–126.

[21] S. Vanstone, Responses to NIST's proposal, Communications of the ACM 35 (July) (1992) 50–52 (communicated by John Anderson).

[22] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory IT-22 (6) (1976) 644–654.

[23] Y. Zhang, W. Liu, W. Lou, Y. Fang, Location-based compromise-tolerant security mechanisms for wireless sensor networks, IEEE JSAC, Special Issue on Security in Wireless Ad Hoc Networks 24 (2) (2006) 247–260.

[24] S. Zhu, S. Setia, S. Jajodia, LEAP: efficient security mechanism for large-scale distributed sensor networks, in: ACM CCS'03, Washington, DC, 27–31 October 2003.

[25] David J. Malan, Matt Welsh, Michael D. Smith, A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography, in: First IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04), Santa Clara, California, October 2004.

[26] Crossbow Technology. Available from: <http://www.xbow.com/>.

[27] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, Comparing elliptic curve cryptography and RSA on 8-bit CPUs, in: CHES'04, 2004.

[28] Atmel Corporation. Available from: <http://www.atmel.com/>.

[29] D.W. Carman, P.S. Kruus, B.J. Matt, Constraints and approaches for distributed sensor network security, NAI Labs Technical Report #00-010, September 2000.

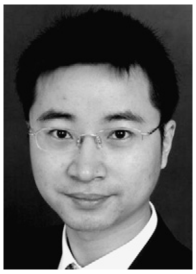
[30] A. Perrig, J. Stankovic, D. Wagner, Security in wireless sensor networks, Communications of the ACM 47 (6) (2004).

[31] D. Liu, P. Ning, Location-based pairwise key establishments for relatively static sensor networks, ACM Workshop on

Security of Ad Hoc and Sensor Networks (SASN'03), October 2003.

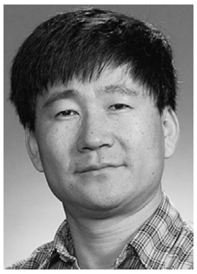
[32] W. Du, J. Deng, Y.S. Han, S. Chen, P.K. Varshney, A key management scheme for wireless sensor networks using deployment knowledge, in: IEEE INFOCOM'04, Hong Kong, March 2004.

[33] Y. Zhou, Y. Zhang, Y. Fang, LLK: a link-layer key establishment scheme in wireless sensor networks, in: IEEE WCNC'05, New Orleans, LA, March 2005.



Yun Zhou received a B.E. degree in electronic information engineering (2000) and an M.E. degree in communication and information system (2003) from the Department of Electronic Engineering and Information Science at the University of Science and Technology of China, Hefei, China. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering at the University of Florida, Gainesville, USA. His research interests are in the areas of security, cryptography, wireless communications and networking, signal processing, and operating system. He is a student member of the IEEE.

Yanchao Zhang received the B.E. degree in Computer Communications from Nanjing University of Posts and Telecommunications, Nanjing, China, in July 1999, and the M.E. degree in Computer Applications from Beijing University of Posts and Telecommunications, Beijing, China, in April 2002. Since September 2002, he has been working towards the Ph.D. degree in the Department of Electrical and Computer Engineering at the University of Florida. His research interests are network and distributed system security, wireless networking, and mobile computing, with emphasis on mobile ad hoc networks, wireless sensor networks, wireless mesh networks, and heterogeneous wired/wireless networks.



Yuguang Fang received a Ph.D. degree in Systems Engineering from Case Western Reserve University in January 1994 and a Ph.D. degree in Electrical Engineering from Boston University in May 1997. He was an assistant professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology from July 1998 to May 2000. He then joined the Department of Electrical and Computer Engineering at University of Florida in May 2000 as an assistant professor, got an early promotion to an associate professor with tenure in August 2003 and a professor in August 2005. He has published over 150 papers in refereed professional journals and conferences. He received the National Science Foundation Faculty Early Career Award in 2001 and the Office of Naval Research Young Investigator Award in 2002. He has served on many editorial boards of technical journals including IEEE Transactions on Communications, IEEE Transactions on Wireless Communications, IEEE Transactions on Mobile Computing and ACM Wireless Networks. He is a senior member of the IEEE.