

RESEARCH ARTICLE

# A robust user authentication scheme with self-certificates for wireless sensor networks

Huei-Ru Tseng, Rong-Hong Jan and Wu Yang\*

Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan

## ABSTRACT

User authentication is a critical part of security, along with confidentiality and integrity, for computer systems that allow legitimate users remote access over an open communication network. Recently, user authentication for wireless sensor networks (WSNs) has received considerable attention. We propose a robust user authentication scheme for WSNs. The scheme is based on elliptic-curve cryptosystems with self-certificates. The proposed scheme allows users to change their key pairs without interaction with a key distribution center (KDC). Moreover, the proposed scheme still works well even if the adversary captures  $t$  nodes out of  $n$  nodes in the WSNs. Security of the proposed scheme is modeled and analyzed with Petri nets. Our analysis shows that the proposed scheme can successfully defend some of the most notorious attacks, including replay attacks, forgery attacks, and node-capture attacks. Copyright © 2010 John Wiley & Sons, Ltd.

## KEYWORDS

authentication; wireless sensor networks; elliptic-curve cryptosystems; self-certificates; Petri nets

### \*Correspondence

Wu Yang, Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan.

E-mail: wuyang@cs.nctu.edu.tw

## 1. INTRODUCTION

Wireless sensor networks (WSNs) consist of spatially distributed sensors used to cooperatively monitor environmental phenomena, such as humidity, temperature, motion, pressure, or vibration. In general, most queries in WSN applications are issued at the base stations or at the backend of the application system. However, in some critical applications, such as military surveillance, it is necessary that all real-time battlefield data could be accessed from every sensor node. In addition, security measures should be taken to prevent enemies from accessing or modifying the collected data, which is highly sensitive.

To control access to WSNs, it is essential for sensor nodes to authenticate the users. Even though a number of user authentication schemes with smart cards [1–7] have been proposed, these existing schemes cannot be directly applied to user authentication in WSNs due to the limited computational power and energy supply in sensor nodes. In 2001, Perrig *et al.* [8] proposed security protocols for WSNs (SPINS), providing important security primitives: authenticated and confidential communication, and authenticated broadcast. They designed an authenticated routing-scheme and a secure node-to-node key agreement protocol. In 2004, user authentication in WSNs was proposed by Benenson *et*

*al.* [9]. Since then, a lot of research work [10–18] has been done in this field.

Compared with symmetric-key cryptography widely used in WSNs, public-key cryptography provides a more flexible interface that requires no complicated key pre-distribution and management as in symmetric-key schemes [13,18]. Over the past few years, elliptic-curve cryptosystem (ECC) has attracted considerable attention as ECC devices have higher strength per key bit, lower power consumption, and smaller bandwidth compared to RSA cryptosystems [19,20]. For example, an elliptic curve over a 163-bit field gives the same level of security as a 1024-bit RSA modulus [20]. In addition, the recent progress in 160-bit ECC implementation shows that an ECC point multiplication takes less than 1 s, which proves that ECC is feasible for resource-constrained platforms such as wireless devices [18,21,22].

As completely preventing any physical captures is a costly option, it is cheaper to design security schemes for WSNs that can tolerate a certain number of node captures [12]. Therefore, we propose a robust user authentication scheme for WSNs based on ECC. This scheme can withstand capture of up to  $t$  sensor nodes. We assume a public-key infrastructure (PKI) for ECC [10,13,14,18,21,22]. There is a key distribution center

(KDC) in WSNs, which has a private/public key pair and is responsible for generating the private/public key pairs for users and sensor nodes. Prior to deployment, each user and sensor node has the public key of the KDC pre-loaded.

The proposed scheme is based on self-certificates, which enable users to generate their own certificates and to change their key pairs without the involvement of the KDC. A self-certificate is first generated by a user  $A$  and is encrypted with  $A$ 's private key. The receiver of the self-certificate verifies the self-certificate with  $A$ 's public key. The receiver can trust  $A$ 's public key because it is endorsed by a trusted third party, such as a KDC.

Additionally, the proposed scheme provides many desired features: (1) it can deal with authenticated queries involving multiple sensor nodes; (2) it achieves mutual authentication and key agreement between users and sensor nodes; (3) it provides a KDC to revoke compromised key pairs. Moreover, Petri nets [23] may be used to infer what an attacker could know if he happens to know certain items in the security protocol. We used Petri nets in the security analysis of the proposed scheme. Our analysis shows that the proposed scheme can successfully defend several notorious attacks, including replay attacks, forgery attacks, and node-capture attacks.

The rest of this paper is organized as follows: In Section 2, we briefly review several existing user authentication schemes for WSNs. Next, we introduce our system model in Section 3. Then, in Section 4, we propose a user authentication scheme for WSNs and analyze the proposed scheme in Section 5. Finally, we conclude this paper in Section 6.

## 2. RELATED WORKS

In 2001, Perrig *et al.* [8] proposed security protocols for WSNs (SPINS), providing important security primitives: authenticated and confidential communication, and authenticated broadcast. They designed an authenticated routing scheme and a secure node-to-node key agreement protocol. User authentication in WSNs was proposed by Benenson *et al.* [9] in 2004. They investigated several security issues in WSNs, including access control, and also introduced the notion of  $(t, n)$ -threshold authentication, which means the authentication succeeds if the user can be successfully authenticated with at least  $(n - t)$  out of  $n$  sensors. The rest of the sensors could be compromised or out of order. Thereafter, Benenson *et al.* [10] proposed the first solution to the user authentication problem in the presence of node-capture attacks. Their scheme is based on public-key cryptography, and is designed for a sensor node to authenticate the users.

In 2006, Banerjee and Mukhopadhyay [12] proposed authenticated querying in WSNs that is based on symmetric keys. The scheme can deal with queries involving multiple sensors. However, identifying the involved sensor nodes and flooding the access requests turn out to be very challenging for WSNs. Later, Wang and Li [13] proposed a distributed user access control mechanism under a realistic adversary model for sensor networks. The scheme, which is based

on ECC, is divided into local authentication, which is conducted by the *local sensors*, that is, those sensors that are located physically close to the user, and remote authentication, which is based on the endorsement of the local sensors.

In order to achieve better performance, Wong *et al.* [15] proposed the first password-based user authentication scheme for WSNs. Compared with earlier works, their scheme is efficient since the protocol participants perform only a few hash operations. Unfortunately, Tseng *et al.* [16] showed that Wong *et al.*'s scheme suffers from vulnerabilities to both replay and forgery attacks and proposed an improved scheme. However, these schemes [15,16] can only solve the access-control problem for individual sensor nodes, but not for the whole sensor networks.

Recently, Jiang *et al.* [14] proposed a user authentication scheme based on the self-certified-key cryptosystem [24] and used ECC to establish pair-wise keys between users and sensor nodes. However, the self-certified-key cryptosystem is not without security flaws. Lee and Kim [25] showed that the self-certified-key cryptosystem cannot provide explicit authentication for the public key. An attacker can produce a seemingly valid self-certified key with a third party's identity. This bogus key cannot be distinguished from a valid one until successful communication with the real owner of the identity. To solve the bogus key problem, they introduced the *self-certificate* for the self-certified key. It is a user-generated certificate for the authentication of the self-certified key.

In this paper, a robust user authentication scheme based on the self-certificate cryptosystem [25] is proposed. We modified the self-certificate cryptosystem [25] to use ECC, which is suitable for WSNs nowadays. The proposed scheme can deal with authenticated queries involving multiple sensor nodes. It achieves mutual authentication and key agreement between users and sensor nodes and allows users to change their private/public key pairs without interaction with a KDC and hence gives users more convenience and security. Moreover, the scheme provides a KDC to revoke compromised key pairs.

## 3. SYSTEM AND ADVERSARY MODEL

We assume a PKI for ECC [10,13,14,18,21,22]. There is a KDC in WSNs, which has a private/public key pair and is responsible for generating the private/public key pairs for users and sensor nodes. Prior to deployment, each user and sensor node has the public key of the KDC pre-loaded. With that public key, each entity can verify the certificates endorsed by the KDC.

In addition, we assume a large static sensor network. Each sensor node is assumed to have the same transmission range and communicates with each other *via* bi-directional wireless channels. A user can send data requests to the sensor nodes within his communication range and receives valid responses if the requests are legitimate. Note that when a

**Table I.** Notations.

Symbol	Definition
$GF(q_1)$	A prime Galois field, where operations are done modulo prime number $q_1$ of length greater than 160-bit
$P$	A base point with order $q_2$
$s$	The KDC's private key
$K_{pub}$	The KDC's public key, where $K_{pub} = s \cdot P$
$ID_i$	User $i$ 's or sensor node $i$ 's identity
$(S_i, Q_i)$	User $i$ 's or sensor node $i$ 's private/public key pair
$CI_i$	The certificate information generated by the KDC
$K_{i,j}$	The pair-wise key computed by the entity $i$ and entity $j$
$COMM_i$	The set of sensor nodes within the communication range of the user $i$
$h(\cdot)$	A one-way hash function
$\parallel$	Concatenation

node of WSNs is physically captured by an adversary, all the secrets stored in that node could be revealed. Because completely preventing any physical captures is a costly option, it is cheaper to design security schemes for WSNs that can tolerate a certain number of node captures [12]. On average, there are  $n$  sensors in the communication range of the user. Of these,  $t$  sensors are allowed to be malicious or to fail. It is assumed that  $t < n/2$ , i.e., the majority of sensors are honest. The assumption is reasonable since compromising sensors takes time and effort. Therefore, the user can rely on communication among at least a half of sensors in his communication range. Our proposed scheme still works well even if the adversary captures  $t$  nodes out of  $n$  nodes in the WSNs. We call the proposed scheme a  $(t, n)$ -threshold authentication scheme.

## 4. PROPOSED SCHEME

In this section, we present a user authentication scheme with self-certificates for WSNs. The proposed scheme is divided into four phases: pre-deployment, login-and-authentication, user-controlled key change, and key revocation. We list the notations in Table I and define a self-certificate in Table II.

**Table II.** Formal definition of a self-certificate.

Let  $(S_i, Q_i)$  be entity (sensor or user)  $i$ 's private/public key pair issued by the KDC, and  $CI_i$  be entity  $i$ 's certificate information. Entity  $i$  signs on  $(CI_i, Q_i)$  with his private key  $S_i$  to generate:  

$$\text{Self-Cert}_i = \text{Sign}_{S_i}(CI_i, Q_i)$$
Then  $\text{Self-Cert}_i$  is called a self-certificate for the public key  $Q_i$ .

### 4.1. The pre-deployment phase

Firstly, the KDC defines an elliptic curve over a prime Galois field  $GF(q_1)$  and chooses a base point  $P$  with order  $q_2$  belonging to this elliptic curve group. Then, it randomly selects a number  $s \in GF(q_2)$  as its private key and performs the point multiplication  $s \cdot P$  on the elliptic curve to compute its public key  $K_{pub}$ .

For every entity (sensor or user)  $i$ , the KDC generates its identity and private/public key pair as follows:

1. Randomly choose  $ID_i \in GF(q_2)$  as entity  $i$ 's identity.
2. Perform the point multiplication  $r_i \cdot P$  to compute  $R_i$ , where  $r_i$  is a random number, i.e.,  $R_i = r_i \cdot P$ .
3. Prepare the certificate information  $CI_i$  as follows:

$$CI_i = [\text{CertNo} \parallel ID_i \parallel ID_{KDC} \parallel R_i \parallel P \parallel K_{pub} \parallel \text{ValidPeriod}](1)$$

where  $\text{CertNo}$  is the certificate serial number and  $\text{ValidPeriod}$  is the valid time period of the certificate.

4. Generate entity  $i$ 's private key  $S_i$  and perform the point multiplication to compute the corresponding public key  $Q_i$  as follows:

$$S_i = s \cdot h(CI_i) + r_i \quad (2)$$

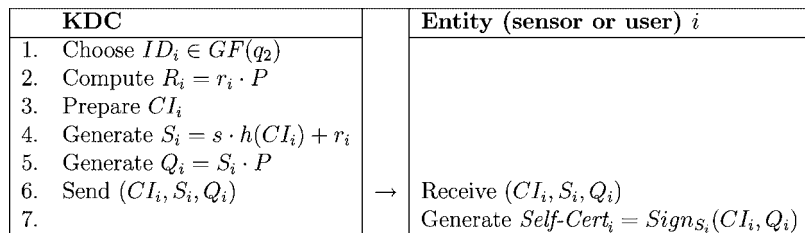
$$Q_i = S_i \cdot P \quad (3)$$

5. Send  $(CI_i, S_i, Q_i)$  to entity  $i$  via a secure channel.

Upon receiving  $(CI_i, S_i, Q_i)$ , entity  $i$  signs  $(CI_i, Q_i)$  with its private key  $S_i$  and generates the self-certificate of the public key  $Q_i$  as follows:

$$\text{Self-Cert}_i = \text{Sign}_{S_i}(CI_i, Q_i) \quad (4)$$

The overall operation of the pre-deployment phase is illustrated in Figure 1.

**Figure 1.** The pre-deployment phase of the proposed scheme.

## 4.2. The login-and-authentication phase

When user  $i$  wishes to query sensor data, he communicates with the sensor nodes within his communication range. The detailed steps are as follows:

1.  $U_i \rightarrow$  WSNs :  $\{CI_i, Q_i, R_i, \text{Self-Cert}_i\}$   
 $U_i$  broadcasts his certificate information  $CI_i$ , public key  $Q_i$ , signature parameter  $R_i$ , and the self-certificate  $\text{Self-Cert}_i$ . Let  $\text{COMM}_i$  denote the set of sensor nodes within the communication range of  $U_i$ .
2. Every  $j \in \text{COMM}_i$  : verify  $Q_i$  and  $\text{Self-Cert}_i$   
 Each sensor node  $j \in \text{COMM}_i$  checks the validity of  $U_i$ 's public key  $Q_i$  and the self-certificate  $\text{Self-Cert}_i$ . Sensor node  $j$  computes  $K_{\text{pub}} \cdot h(CI_i) + R_i$  and checks if  $Q_i = S_i \cdot P$  as follows:  
 Note that

$$\begin{aligned} K_{\text{pub}} \cdot h(CI_i) + R_i &= s \cdot P \cdot h(CI_i) + r_i \cdot P \\ &= (s \cdot h(CI_i) + r_i) \cdot P \\ &= S_i \cdot P \end{aligned} \quad (5)$$

The operations in Equation (5) are performed on the elliptic curve. Sensor node  $j$  then extracts  $CI_i$  and  $Q_i$  from  $\text{Self-Cert}_i$  with the public key  $Q_i$  and checks if  $CI_i$  and  $Q_i$  are correct.

3. Every  $j \in \text{COMM}_i : j \rightarrow U_i :$   
 $\{CI_j, Q_j, R_j, \text{Self-Cert}_j, \text{MAC}_{K_{j,i}}(m_j)\}$   
 If sensor node  $j$  successfully authenticates  $U_i$ , it performs the point multiplication  $S_j \cdot Q_i$  to compute the pair-wise key  $K_{j,i}$ , i.e.,  $K_{j,i} = S_j \cdot Q_i$ . Then, it chooses a random nonce  $m_j$  and calculates the message authentication code (MAC) [26] with  $K_{j,i}$ .
4.  $U_i$  : Verify  $Q_j$  and  $\text{Self-Cert}_j$   
 $U_i$  verifies whether sensor node  $j$ 's public key  $Q_j$  and the self-certificate  $\text{Self-Cert}_j$  are valid. If so, he performs the point multiplication  $S_i \cdot Q_j$  to compute the pair-wise key  $K_{i,j}$ , i.e.,  $K_{i,j} = S_i \cdot Q_j$ .
5.  $U_i \rightarrow$  WSNs : Compute and broadcast  $\{v\}$   
 $U_i$  decrypts the MAC with the corresponding pair-

wise key  $K_{i,j}$  and obtains the nonce  $m'_j$ . This is because:

$$\begin{aligned} K_{i,j} &= S_i \cdot Q_j \\ &= S_i \cdot S_j \cdot P \\ &= Q_j \cdot S_j \\ &= K_{j,i} \end{aligned} \quad (6)$$

The operations in Equation (6) are performed on the elliptic curve. Upon collecting all the nonces, he constructs the authentication value  $v = m'_1 \parallel \dots \parallel m'_n$  and then broadcasts  $\{v\}$ .

6. Every  $j \in \text{COMM}_i$  : Verify  $m_j \in v$   
 Each sensor node  $j \in \text{COMM}_i$  verifies whether  $U_i$  correctly responds to the challenge by checking whether  $m_j$  is in  $v$ . If so, the sensor node broadcasts to other nodes its *yes* vote. Otherwise, it remains silent. If  $(n - t)$  or more *yes* votes are collected, the sensor node believes  $U_i$  is a legitimate user. Note that in some situations, there could be bogus votes. To deal with the bogus-vote problem, the sensor nodes could use the pair-wise keys to encrypt the votes and related information, such as sensor nodes' identities and the timestamps, before broadcasting the encrypted messages.

The overall operation of the login-and-authentication phase is illustrated in Figure 2.

## 4.3. The user-controlled key change phase

A fixed key pair is much easier to attack than a frequently changing one. In certificate-based schemes, changing a key pair usually requires complicated interaction between a user and a KDC. In our scheme, a user can change his key pair without interaction with the KDC.

Let  $(S_i, Q_i)$  be user  $i$ 's private/public key pair issued by the KDC and let  $\text{Self-Cert}_i$  be the self-certificate issued by  $U_i$ . He can generate a new key pair  $(S'_i, Q'_i)$  and a new

User $i$		Sensor node $j \in \text{COMM}_i$
1. Broadcast $\{CI_i, Q_i, R_i, \text{Self-Cert}_i\}$	$\rightarrow$	Receive $\{CI_i, Q_i, R_i, \text{Self-Cert}_i\}$
2.		Verify $Q_i$ and $\text{Self-Cert}_i$
3.		Generate $m_j$
4.		Compute $K_{j,i} = S_j \cdot Q_i$
5.		Compute $\text{MAC}_{K_{j,i}}(m_j)$
6. Receive $\{CI_j, Q_j, R_j, \text{Self-Cert}_j,$	$\leftarrow$	Send $\{CI_j, Q_j, R_j, \text{Self-Cert}_j,$
7. $\text{MAC}_{K_{j,i}}(m_j)\}$		$\text{MAC}_{K_{j,i}}(m_j)\}$
8. Verify $Q_j$ and $\text{Self-Cert}_j$		
9. Compute $K_{i,j} = S_i \cdot Q_j$		
10. Compute $\{v\}$		
11. Broadcast $\{v\}$	$\rightarrow$	Receive $\{v\}$
12.		Verify $m_j \in v$

Figure 2. The login-and-authentication phase of the proposed scheme.

certificate  $\text{Self-Cert}'_i$  with the following operations.

1. Perform the point multiplication  $r'_i \cdot P$  to compute  $R'_i$ , where  $r'_i$  is a random number, i.e.,  $R'_i = r'_i \cdot P$ .
2. Generate a new private key  $S'_i$  and perform the point multiplication to compute the corresponding public key  $Q'_i$  as follows:

$$S'_i = S_i \cdot h(\text{CI}_i \| R'_i) + r'_i \quad (7)$$

$$Q'_i = S'_i \cdot P \quad (8)$$

3. Generate the self-certificate  $\text{Self-Cert}'_i$  by signing  $(\text{CI}_i, Q'_i)$  with his new private key  $S'_i$  as follows:

$$\text{Self-Cert}'_i = \text{Sign}_{S'_i}(\text{CI}_i, Q'_i) \quad (9)$$

Once the new public key  $Q'_i$  and the self-certificate  $\text{Self-Cert}'_i$  are generated,  $U_i$  will broadcast  $\{\text{CI}_i, Q'_i, R'_i, \text{Self-Cert}'_i\}$ . Every sensor node  $j \in \text{COMM}_i$  computes  $K_{\text{pub}} \cdot h(\text{CI}_i) \cdot h(\text{CI}_i \| R'_i) + R_i \cdot h(\text{CI}_i \| R'_i) + R'_i$  and checks if  $Q'_i = S'_i \cdot P$ .

Note that

$$\begin{aligned} & K_{\text{pub}} \cdot h(\text{CI}_i) \cdot h(\text{CI}_i \| R'_i) + R_i \cdot h(\text{CI}_i \| R'_i) + R'_i \\ &= (s \cdot h(\text{CI}_i) \cdot h(\text{CI}_i \| R'_i) \cdot P) + (r_i \cdot h(\text{CI}_i \| R'_i) \cdot P) + R'_i \\ &= (s \cdot h(\text{CI}_i) + r_i) \cdot h(\text{CI}_i \| R'_i) \cdot P + R'_i \\ &= S_i \cdot h(\text{CI}_i \| R'_i) \cdot P + r'_i \cdot P \\ &= (S_i \cdot h(\text{CI}_i \| R'_i) + r'_i) \cdot P \\ &= S'_i \cdot P \end{aligned} \quad (10)$$

The operations in Equation (10) are performed on the elliptic curve. Sensor node  $j$  then extracts  $\text{CI}_i$  and  $Q'_i$  from  $\text{Self-Cert}'_i$  with the public key  $Q'_i$  and checks if  $\text{CI}_i$  and  $Q'_i$  are correct. If both conditions hold, sensor node  $j$  performs Step 3 in the login-and-authentication phase.

#### 4.4. The key revocation phase

When a certified key pair is found compromised, the KDC can revoke it with a *certificate revocation list* (CRL). The KDC publishes CRL containing the serial numbers of all the certificates for the revoked key pair. Anyone who wants to verify a self-certificate should check the CRL first. Once the certificates of the compromised key are revoked, the compromised key can no longer be used to gain access to sensor data. More details on certificate revocation and certificate update can be found in Reference [27].

## 5. ANALYSIS OF OUR SCHEME

In this section, we show that our scheme can resist several notorious attacks. In addition, we provide a comparative study with other user authentication schemes.

**Table III.** Formal definition of a Petri net.

A Petri net is a 5-tuple,  $\text{PN} = (P, T, F, W, M_0)$  where:

- $P = \{P_1, P_2, \dots, P_m\}$  is a finite set of places,
- $T = \{T_1, T_2, \dots, T_n\}$  is a finite set of transitions,
- $F \subseteq (P \times T) \cup (T \times P)$  is a set of arcs (flow relation),
- $W : F \rightarrow \{1, 2, 3, \dots\}$  is a weight function,
- $M_0 : P \rightarrow \{0, 1, 2, 3, \dots\}$  is the initial marking,
- $P \cap T = \emptyset$  and  $P \cup T \neq \emptyset$ .

A Petri net structure  $N = (P, T, F, W)$  without any specific initial marking is denoted by  $N$ .

A Petri net with the given initial marking is denoted by  $(N, M_0)$ .

### 5.1. Security analysis

In this section, we first use Petri nets [23] to model and analyze the proposed scheme. Next, security properties of our scheme will be specified.

#### 5.1.1. Petri net model.

We used a Petri net to model our security scheme. The formal definition of a Petri net [28] is listed in Table III. Petri nets are composed from graphical symbols designating places (shown as circles), transitions (shown as rectangles), and directed arcs (shown as arrows). The places denote (atomic and composite) data items. The transitions denote decryption or decomposition operations. Arcs run between places and transitions.

When a transition fires, a composite data item is decomposed or decrypted, resulting in one or more simpler data items. Since we assume an open network environment, all data items in the transmitted messages are assumed to be public, and are known to the attacker. There will be tokens in the places representing the data items in the transmitted messages initially. From this initial marking, we can infer what an attacker can know eventually. Furthermore, we can also experiment what an attacker can know if he knows additional data items from other sources. The Petri net model is illustrated in Figure 3. The definitions of the places and transitions used in this model are listed in Tables IV and V, respectively. The model is simulated with the HPSim Petri net simulation tool [29].

#### 5.1.2. Security properties.

The security of the proposed scheme is based on the difficulty of the elliptic-curve discrete logarithm problem (ECDLP), which is believed to be unsolvable in polynomial time. Let  $G_1$  be a group of the prime order  $q$  and  $P$  be an arbitrary generator of  $G_1$ . We view  $G_1$  as an additive group. We first specify the mathematical difficult problem used in this paper.

**Definition 1.** *The ECDLP is defined as follows: given  $Q, R \in G_1$ , find an integer  $x \in \mathbb{Z}_q^*$  such that  $R = xQ$ .*

Now we show that our scheme can resist replay attacks, forgery attacks, and node-capture attacks, and also analyze the security property: mutual authentication.

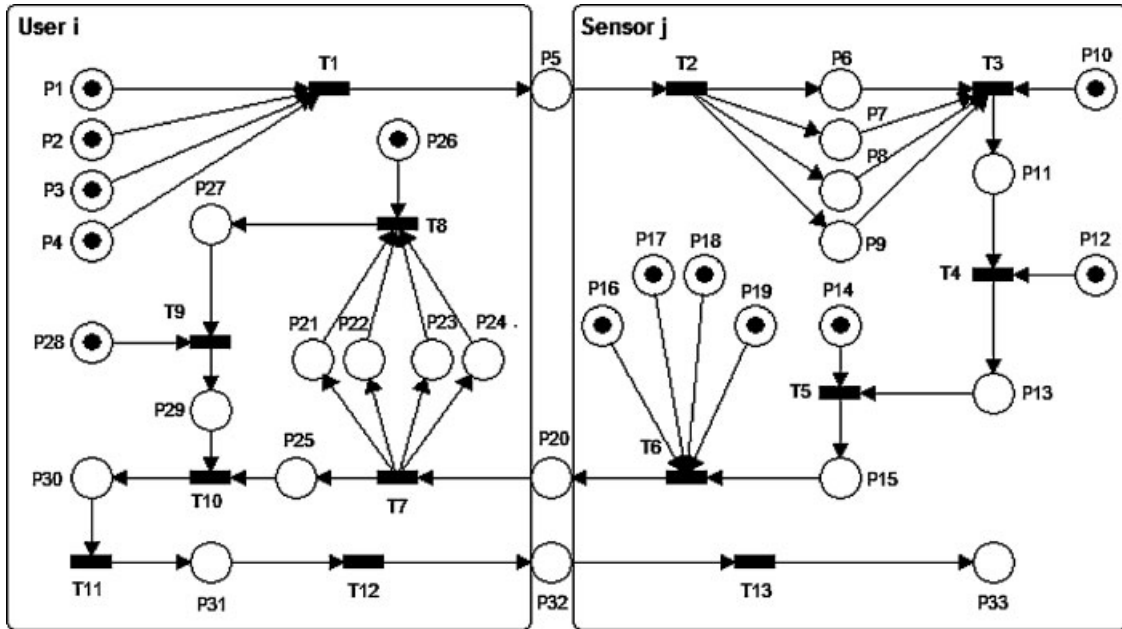


Figure 3. A Petri net model of the proposed scheme.

Table IV. Definitions of places.

Place	Definition	Place	Definition
$P_1$	$Cl_i$	$P_{18}$	$R_j$
$P_2$	$Q_i$	$P_{19}$	Self-Cert <sub>j</sub>
$P_3$	$R_i$	$P_{20}$	Packet{ $Cl_j, Q_j, R_j, \text{Self-Cert}_j, \text{MAC}_{K_{ij}}(m_j)$ }
$P_4$	Self-Cert <sub>i</sub>	$P_{21}$	$Cl_j$
$P_5$	Packet{ $Cl_i, Q_i, R_i, \text{Self-Cert}_i$ }	$P_{22}$	$Q_j$
$P_6$	$Cl_i$	$P_{23}$	$R_j$
$P_7$	$Q_i$	$P_{24}$	Self-Cert <sub>j</sub>
$P_8$	$R_i$	$P_{25}$	$\text{MAC}_{K_{ij}}(m_j)$
$P_9$	Self-Cert <sub>i</sub>	$P_{26}$	$K_{pub}$
$P_{10}$	$K_{pub}$	$P_{27}$	Success verification message
$P_{11}$	Success verification message	$P_{28}$	$S_i$
$P_{12}$	$S_j$	$P_{29}$	$K_{ij}$
$P_{13}$	$K_{ij}$	$P_{30}$	$m'_j$
$P_{14}$	$m_j$	$P_{31}$	$v = m'_1    \dots    m'_n$
$P_{15}$	$\text{MAC}_{K_{ij}}(m_j)$	$P_{32}$	Packet{ $v$ }
$P_{16}$	$Cl_j$	$P_{33}$	Success verification message
$P_{17}$	$Q_j$		

Table V. Definitions of transitions.

Trans.	Definition	Trans.	Definition
$T_1$	Transmit { $Cl_i, Q_i, R_i, \text{Self-Cert}_i$ }	$T_7$	Split the packet
$T_2$	Split the packet	$T_8$	Verify $Q_j$ and Self-Cert <sub>j</sub>
$T_3$	Verify $Q_j$ and Self-Cert <sub>j</sub>	$T_9$	Compute $K_{ij}$
$T_4$	Compute $K_{ij}$	$T_{10}$	Decrypt $\text{MAC}_{K_{ij}}(m_j)$ with $K_{ij}$
$T_5$	Compute $\text{MAC}_{K_{ij}}(m_j)$	$T_{11}$	Compute $v = m'_1    \dots    m'_n$
$T_6$	Transmit { $Cl_j, Q_j, R_j, \text{Self-Cert}_j, \text{MAC}_{K_{ij}}(m_j)$ }	$T_{12}$	Broadcast { $v$ }
		$T_{13}$	Check $m_j \stackrel{?}{=} m'_j$

**Theorem 1.** *The proposed scheme can resist a replay attack.*

*Proof.* Assume an adversary  $A$  eavesdrops the messages  $\{CI_i, Q_i, R_i, \text{Self-Cert}_i\}$  and  $\{v\}$  sent by  $U_i$  and replays them to log in to the system in a later session. Upon receiving the replay message, sensor node  $j$  first verifies  $Q_i$  and  $\text{Self-Cert}_i$ , and then chooses a random nonce  $m_j^*$ . Next,  $j$  computes  $\text{MAC}_{K_{ji}}(m_j^*)$  and sends  $\{CI_j, Q_j, R_j, \text{Self-Cert}_j, \text{MAC}_{K_{ji}}(m_j^*)\}$  back to  $A$ . After receiving the message,  $A$  has to compute  $v^* = m_1'' \parallel \dots \parallel m_n''$  and broadcast  $\{v^*\}$  back to the WSNs. However,  $A$  cannot just replay the message  $\{v\}$  directly since the random nonce  $m_j$  embedded in  $\text{MAC}_{K_{ji}}(m_j)$  is different from  $m_j^*$  in this session. As shown in Figure 3, computing  $m_j$  is defined in transition  $T_{10}$ , which has two input places,  $P_{25}$  and  $P_{29}$ . Place  $P_{25}$  is the value of  $\text{MAC}_{K_{ji}}(m_j)$  and place  $P_{29}$  is the value of  $K_{i,j}$ . Because having no idea about  $K_{i,j}$  to correctly respond the challenge, the adversary cannot launch a replay attack. ■

**Theorem 2.** *The proposed scheme can resist a forgery attack.*

*Proof.* Assume an attacker  $A$  impersonates user  $i$  by submitting  $\{CI_i, Q_i, R_i, \text{Self-Cert}_i\}$  obtained in a previous session. Upon receiving the message, sensor node  $j$  first performs the authentication operations. Then  $j$  sends  $\{CI_j, Q_j, R_j, \text{Self-Cert}_j, \text{MAC}_{K_{ji}}(m_j^*)\}$  back to  $A$ . However,  $A$  cannot decrypt  $\text{MAC}_{K_{ji}}(m_j^*)$  since he does not have user  $i$ 's private key, which is needed for computing the pair-wise key  $K_{i,j}$ . As shown in Figure 3, computing the pair-wise key  $K_{i,j}$  is defined in transition  $T_9$ , which has two input places,  $P_{27}$  and  $P_{28}$ . Place  $P_{28}$  is the value of  $S_i$ . If  $A$  could compute  $U_i$ 's private key somehow, he would have broken the ECDLP as defined in Definition 1. The discrete logarithm problem can be reduced to the problem of computing the private key  $S_i$  from the public key  $Q_i = S_i \cdot P$ . In addition, even if the adversary obtains multiple pair-wise keys  $K_{i,j}$ , it is intractable to compute  $S_i$  due to the hardness of the ECDLP problem. Thus, we claim that computing the private key from the public key and the pair-wise key is at least as difficult as the elliptic-curve discrete logarithm problem. As a result, our scheme is secure against the forgery attacks. ■

**Theorem 3.** *The proposed scheme can resist a node-capture attack.*

*Proof.* It is assumed that  $t < n/2$ , i.e., the majority of sensors are honest. Due to the voting stage in the login-and-authentication phase, if a sensor node can collect at least  $(n - t)$  yes votes, the sensor node believes the user is legitimate. Hence, our scheme can tolerate up to  $t$  nodes being captured. ■

**Theorem 4.** *The proposed scheme can provide mutual authentication.*

*Proof.* The security of the pair-wise key is based on the difficulty of ECDLP, which are believed to be unsolvable

in polynomial time. Using Equation (6), the pair-wise key between  $U_i$  and sensor node  $j$  is established as follows:

$$K_{i,j} = S_i \cdot Q_j = S_i \cdot S_j \cdot P = Q_i \cdot S_j = K_{j,i} \quad (11)$$

As shown in Figure 3, computing a pair-wise key is defined in transition  $T_4$  and transition  $T_9$ . Therefore,  $U_i$  and sensor node  $j$  can use the pair-wise key  $K_{i,j}$  in subsequent communications. ■

## 5.2. Functionality

We summarize the functionality of our proposed scheme in this subsection. The crucial requirements for a user authentication scheme are listed below:

- C1. *(t, n)-threshold authentication:* A scheme can deal with authenticated queries involving multiple sensor nodes and still works well even if the adversary captures  $t$  nodes out of  $n$  nodes in the WSNs.
- C2. *Mutual authentication:* A user and a sensor node can authenticate each other.
- C3. *Key agreement:* After successful authentication, a user and a sensor node mutually agree upon pair-wise keys.
- C4. *User-controlled key change:* A user can change his key pair without interaction with a KDC.
- C5. *Key revokability:* An issued key pair can be revoked, say, when it is found compromised.

We summarize the functionality of related authentication schemes in Table VI.

## 5.3. Efficiency analysis

Now we examine the performance of our proposed scheme. We use the computational and communication overhead as the metric to evaluate the performance of the proposed scheme. Due to the similarity of network scenarios, we com-

**Table VI.** Comparison of user authentication schemes for WSNs.

	C1	C2	C3	C4	C5
Our proposed scheme	Yes	Yes	Yes	Yes	Yes
Benenson <i>et al.</i> 's scheme [10]	No	No	No	No	No
Benenson <i>et al.</i> 's scheme [11]	Yes	No	No	No	No
Banerjee <i>et al.</i> 's scheme [12]	Yes	No	No	No	No
Wang <i>et al.</i> 's scheme [13]	Yes	No	No	No	No
Jiang <i>et al.</i> 's scheme [14]	Yes	Yes	Yes	No	No
Wong <i>et al.</i> 's scheme [15]	No	No	No	No	No
Tseng <i>et al.</i> 's scheme [16]	No	No	No	No	No
Yu <i>et al.</i> 's scheme [17]	No	No	No	No	No

C1:  $(t, n)$ -threshold authentication; C2: mutual authentication; C3: key agreement; C4: user-controlled key change; C5: key revokability.

**Table VII.** Performance comparison in the pre-deployment phase.

Computational type	Jiang <i>et al.</i> 's scheme		Our scheme	
	KDC	Each entity	KDC	Each entity
Random number generation	3	0	3	0
Hash operation	1	0	1	0
Point multiplication	3	0	3	0
Certificate generation*	—	—	0	1

Certificate generation\*: Jiang *et al.*'s scheme [14] provides no certificate generation.

**Table VIII.** Performance comparison in the login-and-authentication phase.

Computational type	Jiang <i>et al.</i> 's scheme		Our scheme	
	Each node	Each user	Each node	Each user
Random number generation	1	0	1	0
Hash operation	1	$n^*$	1 (2)**	$n$
Symmetric encryption	1	0	1 ( $n$ )***	0
Symmetric decryption	0	$n$	0 ( $n$ )***	$n$
Point multiplication	2	$2n$	2 (4)**	$2n$
Certificate verification****	—	—	1	$n$

$n^*$ : Assume there are  $n$  sensors in the communication range of the user.

(2)\*\*: If a changed key is used, it takes one more hash operation and two more point multiplications for each sensor node.

( $n$ )\*\*\*: To deal with the bogus-vote problem, the sensor nodes could use the pair-wise keys to encrypt and decrypt the votes and related information.

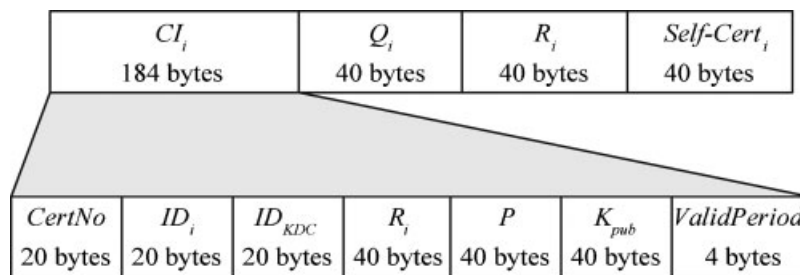
Certificate verification\*\*\*\*: Jiang *et al.*'s scheme [14] does not include certificate verification.

pare our proposed scheme with Jiang *et al.*'s scheme [14], which is presented in Tables VII and VIII. We only compare the computational overhead in two phases (pre-deployment and login-and-authentication) since Jiang *et al.*'s scheme did not include the user-controlled key change and key revocation phases. As illustrated in Table VII, the computational overhead in Jiang *et al.*'s scheme and our scheme in the pre-deployment phase is very similar. The only difference is that each entity needs to generate a self-certificate in our scheme.

As shown in Table VIII, one certificate verification is required for each sensor node during the login-and-authentication phase in our scheme. If a user generates a new key, it takes one more hash operation and two more point multiplications for each sensor node in order to verify the new key. Hence, compared with Jiang *et al.*'s scheme,

our scheme provides various functionalities at the cost of one certificate verification for each sensor node.

The communication overhead is in terms of the following three aspects: the communication overhead incurred by broadcasting the messages from a user to sensors within his transmission range, the overhead incurred by delivering a response from a sensor to a user, and the overhead incurred by transmitting *yes* votes between sensors. In our analysis, we assume a key length of 160 bits in the ECC cryptosystem. As stated in Section 4.2, the user broadcasts  $\{CI_i, Q_i, R_i, Self-Cert_i\}$  in Step 1 and  $\{v\}$  in Step 5. The length of the certificate information  $CI_i$  is 184 bytes, as shown in Figure 4.  $Q_i$  and  $R_i$  each costs 40 bytes. Assume the  $Self-Cert_i$  is constructed by the elliptic-curve digital signature algorithm (ECDSA) [30,31]. The length of the  $Self-Cert_i$  is 40 bytes. Thus, the communication overhead



**Figure 4.** Broadcasting message format from a user to sensors in the login-and-authentication phase.

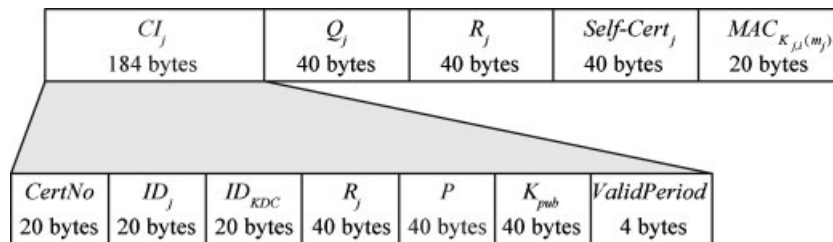


Figure 5. Transmitting message format from a sensor to a user in the login-and-authentication phase.

Table IX. Communication overhead in the login-and-authentication phase of the proposed scheme.

	Each user	Each sensor
Communication overhead	$(304 +  v )$ bytes	$(324 + (n - 1) \times  yes\ vote )$ bytes

$|v|$ \*:  $|v|$  denotes the length of the challenge response sent from a user to sensors.

$(n - 1)$ \*\* : Assume there are  $(n - 1)$  sensors in the communication range of the sensor.

incurred by broadcasting the messages from a user to sensors is  $(304 + |v|)$  bytes.

As stated in Section 4.2, when a sensor transmits  $\{CI_j, Q_j, R_j, Self-Cert_j, MAC_{K_{ji}(m_j)}\}$  to a user in Step 3, as shown in Figure 5, it will cost each sensor 324 bytes. Upon correctly verifying the user, the sensor broadcasts a *yes* vote to other nodes, which costs  $(n - 1) \times |yes\ vote|$  bytes. Note that the sensor nodes could use the pair-wise keys to encrypt the votes and related information to avoid the bogus-vote problem. The total communication overhead is listed in Table IX.

## 6. CONCLUSIONS

In this paper, we proposed a user authentication scheme for WSNs based on ECC. Our scheme is based on self-certificates, which enable users to generate their own certificates. We demonstrated that users can change their key pairs without the involvement of a KDC. Our scheme can also be used to revoke compromised key pairs with a CRL. Moreover, we used Petri nets in the security analysis of the proposed scheme. With our scheme, it is possible to completely prevent adversaries from performing some of the most notorious attacks, such as replay attacks, forgery attacks, and node-capture attacks.

## ACKNOWLEDGEMENTS

This work was supported by the National Science Council, Taiwan, Republic of China, under grant NSC 97-2221-E-009-048-MY3, NSC 97-2221-E-009-049-MY3, and NSC 96-2628-E-009-014-MY3.

## REFERENCES

- Chien HY, Jan JK, Tseng YM. An efficient and practical solution to remote authentication: smart card. *Computers and Security* 2002; **21**(4): 372–375.
- Lee CC, Li LH, Hwang MS. A remote user authentication scheme using hash functions. *ACM SIGOPS Operating Systems Review* 2002; **36**(4): 23–29.
- Juang WS. Efficient password authenticated key agreement using smart cards. *Computers and Security* 2004; **23**(2): 167–173.
- Das ML, Saxena A, Gulati VP. A dynamic id-based remote user authentication scheme. *IEEE Transactions on Consumer Electronics* 2004; **50**(2): 629–631.
- Hsu CL. A user friendly remote authentication scheme with smart cards against impersonation attacks. *Applied Mathematics and Computation* 2005; **170**(1): 135–143.
- Lee Y, Nam J, Kim S, Won D. Two efficient and secure authentication schemes using smart cards. In *Proceedings of International Conference on Computational Science and its Applications (ICCSA 2006)*, May 2006; 858–866.
- Liaw HT, Lin JF, Wu WC. An efficient and complete remote user authentication scheme using smart card. *Mathematical and Computer Modelling* 2006; **44**(1–2): 223–228.
- Perrig A, Szewczyk R, Wen V, Culler D, Tygar JD. SPINS: security protocols for sensor networks. In *Proceedings of International Conference on Mobile Computing and Networking (Mobicom)*, July 2001; 189–199.
- Benenson Z, Gärtner FC, Kesdogan D. User authentication in sensor networks (extended abstract). In *Proceedings of Informatik 2004, Workshop on Sensor Networks*, September 2004.
- Benenson Z, Gedicke N, Raivio O. Realizing robust user authentication in sensor networks. *Workshop on Real-World Wireless Sensor Networks (REALWSN 2005)*, June 2005.
- Benenson Z, Gärtner FC, Kesdogan D. An algorithmic framework for robust access control in wireless sensor

- networks. In *Proceedings of the European Workshop on Wireless Sensor Networks (EWSN 2005)*, January 2005; 158–165.
12. Banerjee S, Mukhopadhyay D. Symmetric key based authenticated querying in wireless sensor networks. In *Proceedings of the First International Conference on Integrated Internet Ad Hoc and Sensor Networks*, May 2006.
  13. Wang H, Li Q. Distributed user access control in sensor networks. In *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2006)*, June 2006; 305–320.
  14. Jiang C, Li B, Xu H. An efficient scheme for user authentication in wireless sensor networks. In *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications Workshops (AINAW 2007)*, May 2007; 438–442.
  15. Wong KHM, Zheng Y, Cao J, Wang S. A dynamic user authentication scheme for wireless sensor networks. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC 2006)*, vol. 1, June 2006; 244–251.
  16. Tseng HR, Jan RH, Yang W. An improved dynamic user authentication scheme for wireless sensor networks. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2007)*, November 2007; 986–990.
  17. Yu S, Ren K, Lou W. FDAC: Toward fine-grained distributed data access control in wireless sensor networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM 2009)*, April 2009; 963–971.
  18. Wang H, Sheng B, Li Q. Elliptic curve cryptography-based access control in sensor networks. *International Journal of Security and Networks* 2006; **1**(3–4): 127–137.
  19. Koblitz N, Menezes A, Vanstone S. The state of elliptic curve cryptography. *Designs, Codes and Cryptography* 2000; **19**(2–3): 173–193.
  20. Lauter K. The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless Communications* 2004; **11**(1): 2004: 62–67.
  21. Malan DJ, Welsh M, Smith MD. A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography. In *Proceedings of the IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, October 2004; 71–80.
  22. Watro R, Kong D, Cuti SF, Gardiner C, Lynn C, Kruus P. TinyPK: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2004; 59–64.
  23. Petri CA. Kommunikation mit Automaten. *Ph. D. Thesis*, University of Bonn, 1962.
  24. Petersen H, Horster P. Self-certified keys—concepts and applications. In *Proceedings of the 3rd Conference on Communications and Multimedia Security*, September 1997.
  25. Lee B, Kim K. Self-certificate: PKI using self-certified key. In *Proceedings of the Conference on Information Security and Cryptology (CISC 2000)*, vol. 10, no. 1, November 2000; 65–73.
  26. Menezes AJ, Oorschot PC, Vanstone SA. Handbook of Applied Cryptography. CRC Press: Boca Raton, Florida, 1997.
  27. Naor M, Nissim K. Certificate revocation and certificate update. *IEEE Journal on Selected Areas in Communications* 2000; **18**(4): 561–570.
  28. Murata T. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, vol. 77, no. 4, April 1989; 541–580.
  29. HPSim 1.1 Petri nets simulation tool, copyright© 1999–2002 Henryk Anschuetz.
  30. ANSI X9.62-2005. Public key cryptography for the financial services industry: the elliptic curve digital signature algorithm (ECDSA). American National Standards Institute, November 2005.
  31. FIPS PUB 186-3. Digital Signature Standard (DSS), Federal Information Processing Standards Publication, June 2009.