Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 139874, 7 pages
http://dx.doi.org/10.1155/2015/139874

Research Article

Robust Distributed Reprogramming Protocol of Wireless Sensor Networks for Healthcare Systems

Kuo-Yu Tsai

Department of Management Information Systems, Hwa Hsia University of Technology, No. 111, Gongzhuan Road, Zhonghe District, New Taipei City 235, Taiwan

Correspondence should be addressed to Kuo-Yu Tsai; kytsai@cc.hwh.edu.tw

Received 26 June 2015; Accepted 20 August 2015

Academic Editor: Sk Md Mizanur Rahman

With the development of the wireless communication technologies, the wireless sensor networks (WSNs for short) are considered as one of the key research areas in healthcare systems. However, there is sometimes a need for removing bugs or adding new functionalities after WSNs are deployed. Wireless reprogramming is a process for propagating a new code image or relevant commands to sensor nodes in WSNs. In this paper, we propose a robust distributed reprogramming protocol of WSNs for healthcare systems, in which the security requirements are proved to be secure based on the elliptic curve discrete logarithm problem, such as existential unforgeability of signature for code image, authenticity of code image, freshness, and node compromised tolerance.

1. Introduction

Wireless sensor networks (WSNs for short) are spatially distributed sensors to gather physical or environmental information. The distributed sensors cooperate to pass their data through the network to a main location [1]. Recently, WSNs are considered as potential applications in providing high quality for healthcare services in recent years [2]. In healthcare applications, it may be necessary for removing bugs or adding new functionalities after WSNs are deployed. The reprogramming is an important operation function to propagate a new code image or relevant commands to sensor nodes in WSNs. However, an adversary may intercept or modify the transmitted messages in a WSN. Hence, it is critical to design a secure protocol for securely propagating a new code image or relevant commands to sensor nodes in WSNs.

In 2008, Das and Joshi [3] adopted orthogonality principle to design a protocol for dynamically updating sensor nodes in WSNs. A shared secret chosen by a base station has to be preinstalled on all sensor nodes before deploying them in a WSN, and the shared secret is used to validate a received advertisement message. After all sensor nodes accept a correct advertisement message, the sensor nodes dynamically update the shared secret. However, Zeng et al. [4] demonstrated that Das and Joshi's protocol is vulnerable to an impersonation attack. Assume that an adversary obtains the shared secret as he/she has compromised a sensor node. Then, the adversary can impersonate the base station to install his/her preferred program on sensor nodes by using the shared secret. Zeng et al. further proposed an improved scheme, but Wang et al. [5] pointed out that Zeng et al.'s proposed protocol [4] still suffers from an impersonation attack. In 2012, He et al. [6] showed that an adversary can impersonate a base station to install his/her preferred program on sensor nodes in a WSN, and further, the adversary can control the whole WSN. He et al. proposed two simple countermeasures which are formally validated by model checking. Soon afterwards, He et al. [7] proposed a secure and distributed reprogramming protocol (SDRP for short) based on an identity-based signature scheme in WSNs. In the SDRP, multiple authorized users are supported and each authorized user may have a different privilege for reprogramming sensor nodes. Subsequently, He et al. [8] and Yeo et al. [9] pointed out that there is a private key compromised problem in the SDRP, respectively. Furthermore, He et al. proposed an improved protocol based on Barreto et al.'s identity-based signature scheme [10]. In 2014, Shim [11] showed

that He et al.'s improved protocol [8] is entirely broken and further proposed an improved protocol based on a pairing-free identity-based signature scheme. However, we find that Shim's proposed improved protocol is unworkable because the signature verification does not hold true. Among various secure protocols in WSNs [3–9, 11–24], most of the published reprogramming protocols [3, 6, 12–14, 16, 18, 20–23] are based on the centralized approach, which assumes the existence of a base station, and only the base station has the authority to reprogram sensor nodes. The centralized approach is not reliable in reality because of inefficiency, weak scalability, and vulnerability to potential attacks along the communication path [24].

Inspired from He et al.'s protocol [7], we adopt a short signature scheme to design a robust distributed reprogramming protocol of WNSs for healthcare applications. In the proposed protocol, each sensor node validates the received message by verifying the corresponding signature. The proposed protocol provides the following requirements.

(1) Existential unforgeability of signature for code image: any adversary cannot successfully forge a signature for any code image by mounting chosen-message attacks. That is, only authorized users can construct valid reprogramming packets.

(2) Authenticity of code image: prior to the start of installation, a sensor node can verify the source of a code image.

(3) Freshness: prior to the start of installation, a sensor node can confirm that the to-be-installed program is the newest version.

(4) Node compromised tolerance: even though a sensor node is compromised, the compromised sensor node will not cause other uncompromised sensor nodes to violate authenticity of code image, integrity of code image, and freshness.

2. Overview of He et al.'s SDRP

In this section, we review the first proposed distributed reprogramming protocol (SDRP for short) proposed by He et al. [7]. The symbols used in the subsequent descriptions are listed as follows.

The Used Symbols and Descriptions

G: an additive group with an order q, where q is a large prime.

G_T: a multiplicative group with an order q, where q is a large prime.

P: a generator in G.

q: a prime number.

$\hat{e}$: a bilinear mapping function, defined as $\hat{e} : G \times G \rightarrow G_T$.

s: a master key for a sensor network owner.

$\mathrm{PK}_{\mathrm{SNO}}$: a public key for a sensor network owner, where $\mathrm{PK}_{\mathrm{SNO}} = s \cdot P$.

H_1: a hash function defined as $H_1 : \{0,1\}^* \rightarrow G$.

H_2: a hash function defined as $H_2 : \{0,1\}^* \rightarrow Z_q^*$.

U_j: an authorized user.

UID_j: an identifier for U_j, where $\mathrm{UID}_j \in \{0,1\}^*$.

Pri_j: a privilege for U_j.

PK_j: a public key for U_j, where $\mathrm{PK}_j = H_1(\mathrm{UID}_j \parallel \mathrm{Pri}_j) \in G$.

SK_j: a private key for U_j, where $\mathrm{SK}_j = s \cdot \mathrm{PK}_j$.

He et al.'s SDRP includes *System Initialization Phase*, *User Preprocessing Phase*, and *Sensor Node Verification Phase*, and there are three kinds of roles: *Sensor Network Owner* (SNO for short), *Authorized User* (AU for short), and *Sensor Node* (SN for short). The detailed descriptions of each phase are given in the following.

System Initialization. In this phase, a sensor network owner SNO first generates his/her own private/public key pair $s/\mathrm{PK}_{\mathrm{SNO}}$ and system parameters. Then, SNO assigns the privilege to each authorized user AU and generates the corresponding key pair for AU. The detailed descriptions of all steps are shown as follows.

(1) Choose an additive group G and a multiplicative group G_T, where G and G_T have the same order q.

(2) Generate a generator P for G.

(3) Choose a bilinear mapping function $\hat{e}$.

(4) Randomly choose a master key $s \in Z_q^*$.

(5) Compute the corresponding public key $\mathrm{PK}_{\mathrm{SNO}} = s \cdot P$.

(6) Choose two secure one-way hash functions H_1 and H_2.

(7) Load the system parameters $\{G, G_T, \hat{e}, q, P, \mathrm{PK}_{\mathrm{SNO}}, H_1, H_2\}$ into the sensor nodes before deploying a sensor network.

(8) Assign the privilege Pri_j to each AU U_j with an identifier $\mathrm{UID}_j \in \{0,1\}^*$.

(9) Generate the corresponding private/public key pair $\mathrm{SK}_j/\mathrm{PK}_j$ for each U_j, where

$$\mathrm{PK}_j = H_1\left(\mathrm{UID}_j \parallel \mathrm{Pri}_j\right) \in G, \qquad \mathrm{SK}_j = s \cdot \mathrm{PK}_j. \tag{1}$$

(10) Send $\{\mathrm{PK}_j, \mathrm{SK}_j, \mathrm{Pri}_j\}$ to each U_j via a secure channel.

User Preprocessing. Suppose that an AU U_j attempts to construct new reprogramming packets. He/she has to sign the packets with his/her private key and then sends them to the sensor nodes. The U_j performs the following tasks.

(1) Partition the code image into l pages with the fixed size.

(2) Split the page i into n packets with the fixed sized, where $1 \leq i \leq l$. Note that the n packets are denoted as $Pkt_{i,1}, APkt_{i,2}, \ldots, Pkt_{i,n}$.

(3) Compute the hashing value for each packet, where the hash value for each packet in the page i is appended to the corresponding packet in the page $i-1$.

(4) Use the Merkle hash tree [25] for authenticating the hash values for the packets in page 1. The packets related to the Merkle hash tree are referred to in page 0. Suppose that m represent the root of the Merkle hash tree and the metadata about the code image.

(5) Sign m to generate the signature σ_j with his/her private key SK_j:

$$\sigma_j = H_2(m) \cdot \mathrm{SK}_j. \tag{2}$$

(6) Send $\{UID_j, \mathrm{Pri}_j, m, \sigma_j\}$ to the target sensor nodes as the notification for the new code image.

Sensor Node Verification. Upon receiving $\{UID_j, \mathrm{Pri}_j, m, \sigma_j\}$, each sensor node SN has to verify the received messages prior to the start of installation as follows.

(1) Verify the legality for the privilege Pri_j and the received messages. If both of them are valid, SN continues to perform Step (2); otherwise, SN rejects the messages.

(2) Verify the signature by using SNO's public key:

$$\hat{e}(\sigma_j, P) = \hat{e}\left(H_2(m) \cdot H_1\left(UID_j \parallel \mathrm{Pri}_j\right), \mathrm{PK}_{\mathrm{SNO}}\right). \tag{3}$$

If the above equality holds, σ_j is the valid signature, and each SN accepts the received messages; otherwise, SN rejects the messages.

Security Analysis. According to He et al.'s [8] and Yeo et al.'s [9] analysis, He et al.'s SDRP [7] is vulnerable to a key compromise attack. Assume that $H_2(m)$ and q are coprime. An adversary can compute the inverse $H_2(m)^{-1}$ and derive the private $\mathrm{SK}_j = H_2(m)^{-1} \cdot \sigma_j$ from the signature σ_j, where $\sigma_j = H_2(m) \cdot \mathrm{SK}_j$. He et al.'s SDRP [7] cannot achieve the security requirements as they claimed.

3. Our Proposed Reprogramming Protocol for Healthcare Systems

The basic idea for our proposed protocol is that maintenance staffs responsible for maintaining different healthcare applications have to register with the sensor owner. Upon the successful registration, a staff receives a smart card storing authorized private key. Then, the staff can construct secure reprogramming packets to WSNs, whenever demanded. In order to assure the validity for the packets, the sensor nodes in WSNs have to verify the signature for the packets.

Our reprogramming protocol also consists of three phases: *System Initialization*, *User Preprocessing*, and *Sensor Node Verification*. The descriptions for *System Initialization Phase* in our proposed protocol are almost the same as ones in He et al.'s SDRP [7], but an extra one-way hash function H_3 is used in our protocol, where $H_3 : G \rightarrow Z_q^*$. The detailed descriptions for *User Preprocessing* and *Sensor Node Verification* are as follows.

User Preprocessing. Assume that an AU U_j constructs the reprogramming packets, signs the packets with his/her own private key, and sends them to the sensor nodes. The steps from (1) to (4) for constructing reprogramming packets are the same as He et al.'s SDRP [7]. We also assume that the message m represents the root of the Merkle hash tree [25] and the metadata about the code image. Other steps for the U_j are as follows.

(5) Choose a random number $r_j \in Z_q^*$.

(6) Compute $R_j = r_j \cdot P$.

(7) Generate the nonce n_j.

(8) Compute S_j by using his/her private key SK_j:

$$S_j = r_j^{-1}\left(H_1\left(m \parallel n_j\right) + H_3\left(R_j\right) \cdot \mathrm{SK}_j\right). \tag{4}$$

(9) Send $\{UID_j, \mathrm{Pri}_j, m, R_j, S_j\}$ to the target sensor nodes as the notification for the new code image.

Sensor Node Verification. Upon receiving $\{UID_j, \mathrm{Pri}_j, m, R_j, S_j\}$, each sensor node SN verifies the received messages to assure the validity prior to the start of installation. Detailed descriptions are as follows.

(1) Verify the legality for the privilege Pri_j and the received message. If both of them are valid, SN continues to perform the next step; otherwise, SN rejects the received messages.

(2) Verify the signature with the public key of the SNO and public system parameters:

$$\hat{e}\left(R_j, S_j\right) = \hat{e}\left(P, H_1\left(m \parallel n_j\right)\right)\hat{e}\left(\mathrm{PK}_{\mathrm{SNO}}, Q\right)^{H_3(R_j)}, \tag{5}$$

where $Q = H_1(UID_j \parallel \mathrm{Pri}_j)$. If the above equality holds, the received signature (R_j, S_j) is valid, and each SN accepts the received messages; otherwise, each SN rejects the received messages.

Correctness. According to the verification equation, each sensor node can verify if the signature (R_j, S_j) is valid:

$$\begin{aligned}
&\hat{e}\left(R_j, S_j\right) \\
&\quad = \hat{e}\left(r_j \cdot P, r_j^{-1}\left(H_1\left(m \parallel n_j\right) + H_3\left(R_j\right) \cdot \mathrm{SK}_j\right)\right) \\
&\quad = \hat{e}\left(P, r_j r_j^{-1}\left(H_1\left(m \parallel n_j\right) + H_3\left(R_j\right) \cdot \mathrm{SK}_j\right)\right) \\
&\quad = \hat{e}\left(P, H_1\left(m \parallel n_j\right) + H_3\left(R_j\right) \cdot \mathrm{SK}_j\right) \\
&\quad = \hat{e}\left(P, H_1\left(m \parallel n_j\right)\right)\hat{e}\left(P, H_3\left(R_j\right) \cdot \mathrm{SK}_j\right) \\
&\quad = \hat{e}\left(P, H_1\left(m \parallel n_j\right)\right)\hat{e}\left(P, \mathrm{SK}_j\right)^{H_3(R_j)} \\
&\quad = \hat{e}\left(P, H_1\left(m \parallel n_j\right)\right)\hat{e}\left(P, s \cdot \mathrm{PK}_j\right)^{H_3(R_j)} \\
&\quad = \hat{e}\left(P, H_1\left(m \parallel n_j\right)\right)\hat{e}\left(s \cdot P, \mathrm{PK}_j\right)^{H_3(R_j)}
\end{aligned}$$

$$= \widehat{e}\left(P, H_1\left(m \parallel n_j\right)\right)\widehat{e}\left(\mathrm{PK}_{\mathrm{SNO}}, \mathrm{PK}_j\right)^{H_3(R_j)}$$
$$= \widehat{e}\left(P, H_1\left(m \parallel n_j\right)\right)\widehat{e}\left(\mathrm{PK}_{\mathrm{SNO}}, Q\right)^{H_3(R_j)}. \quad (6)$$

Note that $Q = H_1(\mathrm{UID}_j \parallel \mathrm{Pri}_j)$.

4. Security Analysis and Efficiency Comparison

Based on the elliptic curve discrete logarithm problem (ECDLP for short) [26, 27], the computation Diffie-Hellman problem (CDH for short) [28], the decisional Diffie-Hellman problem (DDH for short) [28], and the assumption of one-way hash function (OWHF for short) [29], we will analyze our reprogramming protocol and verify if it satisfies the security requirements: *existential unforgeability of signature for code image*, *authenticity of code image*, *freshness*, and *node compromise tolerance*. In addition, we will compare the computation costs in our proposed protocol with the ones in He et al.'s protocol [8]. The descriptions for the ECDLP, the CDH, the DDH, and the OWHF are as follows.

Definition 1 (elliptic curve discrete logarithm problem). Let E be an elliptic curve defined over $GF(p)$. Given (P, Q), determining an integer r such that $Q = r \cdot P$ is computationally infeasible, where $P, Q \in E$ the order of E is a large prime q, and $0 \le r \le q - 1$.

Definition 2 (computation Diffie-Hellman problem). Given $(P, a \cdot P, b \cdot P)$, computing $ab \cdot P$ is computationally infeasible, where $P \in G_1$ and $a, b \in Z_q^*$.

Definition 3 (decisional Diffie-Hellman problem). Given $(P, a \cdot P, b \cdot P, c \cdot P)$, deciding if $c = ab$ is computationally infeasible, where $P \in G_1$ and $a, b, c \in Z_q^*$.

Definition 4 (one-way hash function). Let H be a one-way hash function. The input for H is a message m with an arbitrary length, and the output for H is a hashing value $H(m)$ with a fixed length. There are three properties. (1) Given $H(m)$, deriving m is computationally infeasible. (2) Finding two different messages m and m' such that $H(m) = H(m')$ is computationally infeasible. (3) There exists an efficient algorithm to compute $H(m)$.

We give a formal definition for existential unforgeability of a signature for a code image under a chosen-message attack, which is based on the established notion of existential unforgeability against chosen-message attacks [30, 31]. The game between an adversary $\mathscr{A}$ and a challenger $\mathscr{C}$ is defined as follows.

S: given a security parameter k, the challenger $\mathscr{C}$ first builds up a system parameter set π. Then, the $\mathscr{C}$ generates a private/public key pair $(s, \mathrm{PK}_{\mathrm{SNO}})$ and a private/public key pair $(\mathrm{SK}_j, \mathrm{PK}_j)$ for each authorized user AU with an identifier U_j. Finally, the adversary $\mathscr{A}$ is given π, $\mathrm{PK}_{\mathrm{SNO}}$, and PK_j, and the private keys SK_j and s are kept secret by the $\mathscr{C}$.

Q: the $\mathscr{A}$ can submit various queries to the $\mathscr{C}$ as follows.

(1) H_1 *queries*: to respond to the $\mathscr{A}$'s queries, the $\mathscr{C}$ maintains a list_{H_1}.

(2) H_3 *queries*: to respond to the $\mathscr{A}$'s queries, the $\mathscr{C}$ maintains a list_{H_3}.

(3) *Signature for code image*: for requesting a signature σ_j for a code image, the $\mathscr{A}$ submits a message m and a public key PK_j. Upon receiving the request, the $\mathscr{C}$ computes the signature under the public key PK_j and returns the result to $\mathscr{A}$.

F: the $\mathscr{A}$ outputs a signature σ^* for a target message m^* and the $\mathscr{A}$ never submits the target message m^* for requesting *signature for code image*. We can say that $\mathscr{A}$ wins the game if the signature σ^* is valid for the target m^*.

Definition 5. The proposed protocol is said to achieve existential unforgeability of signature for code image against chosen-message attacks if successful probability for any polynomially bounded adversary will be negligible in the above game.

Theorem 6. *The proposed protocol achieves existential unforgeability of signature for code image against chosen-message attacks in the random oracle model, assuming the hardness of the CDH problem.*

Proof. Assume that a polynomial-time algorithm $\mathscr{A}$ can forge the signature under a chosen-message attack, and let H_1 and H_3 be random oracles in the simulation and the polynomial-time algorithm $\mathscr{A}$ can make q_{H_1} and q_{H_3} queries to H_1 and H_3, respectively.

We will show how to construct a polynomial-time algorithm β that solves the CDH problem by using the $\mathscr{A}$. Let $(P, a \cdot P, b \cdot P)$ be a random instance of the CDH problem. Computing $ab \cdot P$ is the goal for the β.

In the **S** phase, the algorithm β builds up a system parameter set $\pi = \langle q, G, G_T, \widehat{e}, P, H_1, H_3\rangle$, $I_1 = a \cdot P$, and $I_2 = b \cdot P$ with a security parameter k. Then, the β generates a private/public key pair $(a, \mathrm{PK}_{\mathrm{SNO}})$ for the sensor network owner SNO and a private/public key pair $(\mathrm{SK}_j, \mathrm{PK}_j)$ for an authorized user U_j, where $\mathrm{PK}_{\mathrm{SNO}} = a \cdot P$ and $\mathrm{PK}_j = b \cdot P$. Finally, the $\mathscr{A}$ is given π, $\mathrm{PK}_{\mathrm{SNO}}$, and PK_j, and the private keys a and SK_j are kept secret.

In the **Q** phase, the $\mathscr{A}$ adaptively submits the following queries.

(1) H_1 *queries:* the β has to maintain a list_{H_1} storing $(m, n_j, h_{1,j})$ for responding to the $\mathscr{A}$'s queries. Upon receiving an H_1 query for any subdocument m, the β has to check if there is the tuple $(m, n_j, h_{1,j})$ in list_{H_1}. If it exits in list_{H_1}, the β returns $h_{1,j}$ and terminates the step; otherwise, the β generates a random number $h_{1,j} \in Z_q^*$, returns it, and stores $(m, n_j, h_{1,j})$ in list_{H_1}.

(2) H_3 *queries*: the β has to maintain a list_{H_3} storing $(r_i, R_i, h_{3,i})$ for responding to the $\mathscr{A}$'s queries. Upon receiving an h_3 query for any R_j, the β checks if there is $(r_j, R_j, h_{3,j})$ in list_{H_3}. If the tuple exists in list_{H_3}, the

β returns $h_{3,j}$ and terminates the step; otherwise, the β generates a random number $h_{3,j} \in Z_q^*$, returns it, and stores $(r_j, R_j, h_{3,j})$ in list_{H_3}.

(3) *Signature for code image*: the $\mathcal{A}$ submits m to the β and requests a signature σ_j. Upon receiving the request, the β simulates an H_1 oracle to get $(m, n_j, h_{1,j})$ and simulates an H_3 oracle to get $(r_j, R_j, h_{3,j})$. After the simulation, the β generates a random bit $b_j \in \{0, 1\}$. If $b_j = 1$, the β computes S_i by using SK_j and returns it; otherwise, the β randomly generates S_i and returns it.

Let ε' be the probability that the $\mathcal{A}$ successfully outputs a signature σ_j^* for a target m_j^*. We can say that the successful probability ε that β solves the CDH problem is greater than ε' according to the above simulation. That is, the proposed protocol can achieve *existential unforgeability of signature for code image*. □

Theorem 7. *The proposed protocol is said to achieve authenticity of code image if successful probability for any polynomially bounded adversary will be negligible.*

Proof. The source of a code image must be verified by a sensor node prior to the installation. Assume that an adversary attempts to construct reprogramming packets. Thus, a polynomial-time algorithm $\mathcal{A}$ can simulate the following attacks:

(1) Given a SNO's public key $(a, \text{PK}_{\text{SNO}})$, an AU U_j's identifier $\text{UID}_j \in \{0, 1\}^*$, and the corresponding privilege Pri_j, the polynomial-time algorithm $\mathcal{A}$ attempts to output the private key SK_j for the U_j. First of all, the $\mathcal{A}$ computes the U_j's public key $\text{PK}_j = H_1(\text{UID}_j \parallel \text{Pri}_j)$. Then, the $\mathcal{A}$ derives SNO's private key a from the public key $\text{PK}_{\text{SNO}} = a \cdot P$. Finally, the $\mathcal{A}$ outputs the U_j's private key $\text{SK}_j = s \cdot \text{PK}_j$. However, the $\mathcal{A}$ cannot successfully output the U_j's private key SK_j because the $\mathcal{A}$ cannot derive s from the public key $\text{PK}_{\text{SNO}} = a \cdot P$ assuming the hardness of the ECDLP.

(2) Given a signature (R_j, S_j), an AU U_j's identifier $\text{UID}_j \in \{0, 1\}^*$, and the corresponding privilege Pri_j, the polynomial-time algorithm $\mathcal{A}$ attempts to output the U_j's private key SK_j from the signature. First of all, the $\mathcal{A}$ derives r_j from R_j. Then, the $\mathcal{A}$ computes $r_j \cdot S_j = H_1(m \parallel n_j) + H_3(R_j) \cdot \text{SK}_j$. Finally, the $\mathcal{A}$ computes the inverse $H_3(R_j)^{-1}$ for $H_3(R_j)$ and further outputs the private key SK_j. However, the $\mathcal{A}$ cannot successfully output the U_j's private key SK_j because the $\mathcal{A}$ cannot derive r_j from R_j due to the ECDLP.

According to the above analysis, the successful probability for the polynomial-time algorithm $\mathcal{A}$ cannot successfully be negligible. That is, any adversary cannot successfully impersonate the U_j to construct new reprogramming packets in the proposed protocol. □

Theorem 8. *The proposed protocol is said to achieve freshness if successful probability for any polynomially bounded adversary will be negligible.*

Table 1: Computation comparisons.

	He et al.'s protocol [8]	Our protocol
User preprocessing	$T_{\text{exe}} + T_{\text{ecm}}$	$T_{\text{inv}} + 3T_{\text{ecm}}$
Sensor node verification	$T_{\text{exe}} + T_{\text{mul}} + T_{\text{ecm}} + T_{\text{eca}} + T_{\text{pairing}}$	$T_{\text{exe}} + 3T_{\text{pairing}}$

Proof. In the simulation, a polynomial-time algorithm $\mathcal{A}$ can obtain the same messages shown in Theorem 6. Given a nonce n_j^*, the $\mathcal{A}$ cannot successfully output another nonce n_j' such that $H_1(m \parallel n_j^*) = H_1(m \parallel n_j')$ due to the assumption of the OWHF. That is, the nonce n_j in the proposed protocol is used to ensure freshness. □

Theorem 9. *The proposed protocol is said to achieve node compromise tolerance if successful probability for any polynomially bounded adversary will be negligible.*

Proof. Upon compromising a sensor node, a polynomial-time algorithm $\mathcal{A}$ can obtain the system parameters $\{G, G_T, \hat{e}, q, P, \text{PK}_{\text{SNO}}, H_1, H_3\}$. Further, the polynomial-time algorithm $\mathcal{A}$ can submit the same queries shown in Theorem 6. According to the above analysis, the $\mathcal{A}$ cannot cause an uncompromised node to violate the aforementioned security requirements. □

In 2012, Gouvêa et al. [32] present a software implementation for an elliptic curve cryptosystem and a pairing-based cryptosystem for the MSP430 microcontroller family, which is used in wireless sensor nodes. That is, it is practical for the pairing computation using wireless sensor nodes. Further, we do not consider the time for computing a hashing value. The descriptions of symbols for different operations are shown as follows.

The Symbols for Different Operations.

T_{exe}: the time for computing a modular exponential operation.

T_{mul}: the time for computing a modular multiplication operation.

T_{inv}: the time for computing an inverse operation.

T_{ecm}: the time for computing an elliptic curve multiplication operation.

T_{eca}: the time for computing an elliptic curve addition operation.

T_{pairing}: the time for computing a bilinear function.

In the following, we will compare the computation costs in our proposed protocol with ones in He et al.'s protocol [8]. From Table 1, we can see that our proposed protocol is slightly outperformed and achieves additional security requirements, which is lack of in He et al.'s protocol [8].

5. Conclusions

We propose a robust distributed reprogramming protocol providing existential unforgeability of signature for code

image, authenticity of code image, freshness, and node compromised tolerance. The proposed protocol is applicable to healthcare systems, in which maintenance staffs responsible for maintaining different healthcare applications can construct secure reprogramming packets to WSNs, and the sensor nodes in WSNs can ensure the validity of the constructed packets. Considering the resource limitation of sensor node in WSNs, our future work is to design a lightweight distributed reprogramming protocol in WSNs.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported in part by Taiwan Information Security Center (TWISC), Ministry of Science and Technology (MOST), under the grants 103-2221-E-146-005-MY2 and 103-2221-E-011-090-MY2.

References

[1] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351–367, 2004.

[2] H. Alemdar and C. Ersoy, "Wireless sensor networks for healthcare: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2688–2710, 2010.

[3] M. L. Das and A. Joshi, "Dynamic program update in wireless sensor networks using orthogonality principle," *IEEE Communications Letters*, vol. 12, no. 6, pp. 471–473, 2008.

[4] P. Zeng, Z. Cao, K.-K. R. Choo, and S. Wang, "Security weakness in a dynamic program update protocol for wireless sensor networks," *IEEE Communications Letters*, vol. 13, no. 6, pp. 426–428, 2009.

[5] W. Wang, L. Hu, and Y. Li, "Security analysis of a dynamic program update protocol for wireless sensor networks," *IEEE Communications Letters*, vol. 14, no. 8, pp. 782–784, 2010.

[6] D. He, S. Chan, C. Chen, and J. Bu, "Secure and efficient dynamic program update in wireless sensor networks," *Security and Communication Networks*, vol. 5, no. 7, pp. 823–830, 2012.

[7] D. He, C. Chen, S. Chan, and J. Bu, "SDRP: a secure and distributed reprogramming protocol for wireless sensor networks," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4155–4163, 2012.

[8] D. He, C. Chen, S. Chan, J. Bu, and L. T. Yang, "Security analysis and improvement of a secure and distributed reprogramming protocol for wireless sensor networks," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 11, pp. 5348–5354, 2013.

[9] S. L. Yeo, W.-S. Yap, J. K. Liu, and M. Henricksen, "Comments on analysis and improvement of a secure and efficient handover authentication based on bilinear pairing functions," *IEEE Communications Letters*, vol. 17, no. 8, pp. 1521–1523, 2013.

[10] P. S. L. M. Barreto, B. Libert, N. McCullagh, and J.-J. Quisquater, "Efficient and provably-secure identity-based signatures and signcryption from bilinear maps," in *Advances in Cryptology—ASIACRYPT 2005*, vol. 3788 of *Lecture Notes in Computer Science*, pp. 515–532, Springer, Berlin, Germany, 2005.

[11] K.-A. Shim, "S^2DRP: secure implementations of distributed reprogramming protocol for wireless sensor networks," *Ad Hoc Networks*, vol. 19, pp. 1–8, 2014.

[12] N. Bui, O. Ugus, M. Dissegna, M. Rossi, and M. Zorzi, "An integrated system for secure code distribution in wireless sensor networks," in *Proceedings of the 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM '10)*, pp. 575–581, Mannheim, Germany, April 2010.

[13] I. Doh, J. Lim, and K. Chae, "Code updates based on minimal backbone and group key management for secure sensor networks," *Mathematical and Computer Modelling*, vol. 57, no. 11-12, pp. 2801–2813, 2013.

[14] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the deluge network programming system," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 326–333, Nashville, Tenn, USA, April 2006.

[15] D. He, N. Kumar, J. Chen, C.-C. Lee, N. Chilamkurti, and S.-S. Yeo, "Robust anonymous authentication protocol for healthcare applications using wireless medical sensor networks," *Multimedia Systems*, vol. 21, no. 1, pp. 49–60, 2015.

[16] S. Hyun, P. Ning, A. Liu, and W. Du, "Seluge: secure and DoS-resistant code dissemination in wireless sensor networks," in *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN '08)*, pp. 445–456, St. Louis, Mo, USA, 2008.

[17] P. Kumar, S.-G. Lee, and H.-J. Lee, "E-SAP: efficient-strong authentication protocol for healthcare applications using wireless medical sensor networks," *Sensors*, vol. 12, no. 2, pp. 1625–1647, 2012.

[18] Y. W. Law, Y. Zhang, J. Jin, M. Palaniswami, and P. Havinga, "Secure rateless deluge: pollution-resistant reprogramming and data dissemination for wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, Article ID 685219, 22 pages, 2011.

[19] X. Li, J. Niu, S. Kumari, J. Liao, W. Liang, and M. K. Khan, "A new authentication protocol for healthcare applications using wireless medical sensor networks with user anonymity," *Security and Communication Networks*, 2015.

[20] C. H. Lim, "Secure code dissemination and remote image management using short-lived signatures in WSNs," *IEEE Communications Letters*, vol. 15, no. 4, pp. 362–364, 2011.

[21] R. C. Luo and O. Chen, "Mobile sensor node deployment and asynchronous power management for wireless sensor networks," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 5, pp. 2377–2385, 2012.

[22] C. F. C. De La Parra and J. A. Garcia-Macias, "A protocol for secure and energy-aware reprogramming in WSN," in *Proceedings of the ACM International Wireless Communications and Mobile Computing Conference (IWCMC '09)*, pp. 292–297, Leipzig, Germany, June 2009.

[23] H. Tan, J. Zic, S. K. Jha, and D. Ostry, "Secure multihop network programming with multiple one-way key chains," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 16–31, 2011.

[24] R. Zhang, Y. Zhang, and K. Ren, "DP^2AC: distributed privacy-preserving access control in sensor networks," in *Proceedings of the 28th Conference on Computer Communications (INFOCOM '09)*, pp. 1251–1259, IEEE, Rio de Janeiro, Brazil, April 2009.

[25] R. C. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the IEEE Symposium on Security and Privacy (SP '80)*, pp. 122–134, Oakland, Calif, USA, April 1980.

[26] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society Lecture Note Series, Cambridge University Press, Cambridge, UK, 1999.

[27] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic, Norwell, Mass, USA, 1993.

[28] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology—ASIACRYPT 2001*, vol. 2248 of *Lecture Notes in Computer Science*, pp. 514–532, Springer, Berlin, Germany, 2001.

[29] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, New York, NY, USA, 2nd edition, 1996.

[30] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, 1988.

[31] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.

[32] C. P. L. Gouvêa, L. B. Oliveira, and J. López, "Efficient software implementation of public-key cryptography on sensor networks using the MSP430X microcontroller," *Journal of Cryptographic Engineering*, vol. 2, no. 1, pp. 19–29, 2012.