Security Enhanced User Authentication Protocol for Wireless Sensor Networks Using Elliptic Curves Cryptography

Younsung Choi 1, Donghoon Lee 1, Jiye Kim 1, Jaewook Jung 1, Junghyun Nam 2 and Dongho Won 1,*

1 College of Information and Communication Engineering, Sungkyunkwan University, Jangangu, Suwonsi, Gyeonggido 440-746, Korea; E-Mails: yschoi@security.re.kr (Y.C.); dhlee@security.re.kr (D.L.); jykim@security.re.kr (J.K.); jwjung@security.re.kr (J.J.)
2 Department of Computer Engineering, Konkuk University, 268 Chungwondaero, Chungju, Chungcheongbukdo 380-701, Korea; E-Mail: jhnam@kku.ac.kr

* Author to whom correspondence should be addressed; E-Mail: dhwon@security.re.kr; Tel.: +82-31-290-7213; Fax: +82-31-290-7686.

Received: 11 March 2014; in revised form: 29 May 2014 / Accepted: 1 June 2014 / Published: 10 June 2014

Abstract: Wireless sensor networks (WSNs) consist of sensors, gateways and users. Sensors are widely distributed to monitor various conditions, such as temperature, sound, speed and pressure but they have limited computational ability and energy. To reduce the resource use of sensors and enhance the security of WSNs, various user authentication protocols have been proposed. In 2011, Yeh et al. first proposed a user authentication protocol based on elliptic curve cryptography (ECC) for WSNs. However, it turned out that Yeh et al.’s protocol does not provide mutual authentication, perfect forward secrecy, and key agreement between the user and sensor. Later in 2013, Shi et al. proposed a new user authentication protocol that improves both security and efficiency of Yeh et al.’s protocol. However, Shi et al.’s improvement introduces other security weaknesses. In this paper, we show that Shi et al.’s improved protocol is vulnerable to session key attack, stolen smart card attack, and sensor energy exhausting attack. In addition, we propose a new, security-enhanced user authentication protocol using ECC for WSNs.

Keywords: authentication protocol; elliptic curves cryptography; wireless sensor network
1. Introduction

Wireless sensor networks (WSNs) provide a feasible real-time monitoring system. Wireless sensors can be easily deployed in various environments such as military surveillance, forest fire detection, health care and wildlife monitoring. WSNs basically consist of users, sensors and gateways whose communication security is a significant concern in real-world applications [1]. Users and gateways have sufficient resources to be used in the system, but sensors are different. Sensors have limited computational ability, low battery, low bandwidth, and a small amount of memory. Therefore, in WSNs, it is important to reduce the use of sensors to extend their lifespans [2–4].

Various user authentication protocols have been proposed for securing WSNs while minimizing the use of sensors. In 2004, Watro et al. proposed a user authentication protocol employing the RSA and Diffie-Hellman algorithms [5]. In 2006, Wong et al. proposed an efficient dynamic user authentication protocol using a hash function [6]. However, Tseng et al. demonstrated that Wong et al.’s authentication protocol is vulnerable to stolen-verifier attack, replay attack and forgery attack [7,8]. Later in 2009, Das proposed a two-factor user authentication protocol using smart cards. Das showed how to design an authentication protocol where only the user who is in possession of both the smart card and the password can pass the verification of the gateway [8]. However, several security-related flaws in Das’s protocol have been disclosed by later studies as summarized below:

- He et al. demonstrated that Das’s protocol is vulnerable to insider attacks and impersonation attacks, and that it does not allow users to change their passwords freely. He et al. proposed an improved two-factor protocol [9] which can resist insider and impersonation attacks.
- Khan and Alghathbar showed that Das’s protocol fails to provide mutual authentication between the gateway and the sensor, and due to this failure, it is not secure against a gateway bypassing attack and a privileged-insider attack [10].
- Chen et al. also pointed out that Das’s protocol does not achieve mutual authentication between the gateway and the sensor, and proposed a robust authentication protocol that provides the property of mutual authentication [11].

In 2011, Yeh et al. [2] revealed that Chen et al.’s protocol has difficulty in updating users’ passwords and is vulnerable to an insider attack. As an improvement of Chen et al.’s protocol, Yeh et al. presented the first user authentication protocol that uses elliptic curve cryptography (ECC) in WSN environments. However, Han [12] showed that Yeh et al.’s protocol has still some security weaknesses: it does not provide perfect forward secrecy and fails to achieve mutual authentication and key agreement between the user and the sensor. To address these problems with Yeh et al.’s protocol, Shi et al. [3] have recently proposed a new smart-card-based user authentication protocol using ECC for WSNs. Shi et al.’s protocol performs more efficiently, both in terms of computation and communication costs, and provides better security than Yeh et al.’s protocol. However, we found that Shi et al.’s improvement is not secure enough yet and their protocol is susceptible to session key attacks, stolen smart card attacks, and sensor energy exhausting attacks. In addition to reporting the security weaknesses, we also show how to enhance the security of Shi et al.’s protocol with no significant increase in communication and computation costs. We analyze and verify the security of the proposed protocol using non-monotonic cryptographic logic (Rubin logic).
Throughout the paper, we make the following assumptions on the capabilities of the probabilistic polynomial-time adversary $A$ in order to properly capture security requirements of two-factor authentication protocols using smart cards in wireless sensor networks.

- $A$ has the complete control of all message exchanges between the protocol participants: a user, a sensor and the gateway. That is, $A$ can eavesdrop, insert, modify, intercept, and delete messages exchanged among the three parties at will.
- $A$ is able to (1) extract the sensitive information on the smart card of a user through a power analysis attack [13,14] or (2) find out the user's password possibly via shoulder-surfing or by employing a malicious card reader. However, it is assumed that $A$ is unable to compromise both the two factors: the information on the smart card and the password of the user; it is clear that there is no way to prevent $A$ from impersonating the user if both factors are compromised.

2. Overview of Elliptic Curves Cryptography

In 1985, Neal Koblitz and Victor S. Miller proposed the use of elliptic curves in cryptography. After various studies on ECC, it has been widely used since the early 21st century. ECC is a type of public-key cryptography and based on the algebraic structure of elliptic curves over finite fields. Elliptic curves are also used in several integer factorization algorithms. ECC provides the important benefit of a smaller key size, despite which it is able to maintain the same degree of security as other types of public-key cryptography, such as RSA, DH and DSA. Therefore, ECC is especially useful for wireless devices, which typically have limited CPU capacity, power and network connectivity. Table 1 shows the NIST guidelines on choosing key sizes in ECC and other public key cryptography [15].

| Security (bits) | ECC | RSA/DH/DSA | MIPS-Years to Attack | Protection Lifetime |
|----------------|-----|------------|----------------------|---------------------|
| 80             | 160 | 1,024      | $10^{12}$            | until 2010          |
| 112            | 224 | 2,048      | $10^{24}$            | until 2030          |
| 128            | 256 | 3,072      | $10^{28}$            | beyond 2031         |
| 192            | 384 | 7,680      | $10^{47}$            | beyond 2031         |
| 256            | 512 | 15,360     | $10^{66}$            | beyond 2031         |

ECC has three related mathematical problems: the Elliptic Curve Discrete Logarithm Problem (ECDLP), Elliptic Curve Computational Diffie-Hellman Problem (ECCDHP), and Elliptic Curve Decisional Diffie-Hellman Problem (ECDDHP). No polynomial time algorithm can solve the ECDLP, ECCDHP and ECDDHP with non-negligible probability.

Let $p > 3$ be a large prime and choose two field elements $a, b \in \mathbb{F}_p$ satisfying $4a^3 + 27b^2 \neq 0 \mod p$ to define the equation of a non-supersingular elliptic curve $E$: $y^2 = x^3 + ax + b \mod p$ over $\mathbb{F}_p$. Choose a generator point $P = (x_P, y_P)$ whose order is a large prime number $q$ over $E(\mathbb{F}_p)$. In the same way, a subgroup $G$ of the elliptic curve group $E(\mathbb{F}_p)$ with order $q$ is constructed. Then, the three mathematical problems in ECC are defined at various study [16–18] as follows.
• ECDLP: Given a point element $Q$ in $G$, find an integer $x \in \mathbb{Z}_q^*$ such that $Q = xP$, where $xP$ indicates that the point $P$ is added to itself $x$ times by the elliptic curves operation.
• ECCDHP: For $a, b \in \mathbb{Z}_q^*$, given two point elements $aP, bP$ in $G$, compute $abP$ in $G$.
• ECDDHP: For $a, b, c \in \mathbb{Z}_q^*$, given three point elements $aP, bP$ and $cP$ in $G$, decide whether $cP = abP$.

3. Review of Shi et al.’s Protocol

In Shi et al.’s protocol [3], the gateway is a trusted node that holds two sufficiently large master keys, $x$ and $y$. Before starting the system, the gateway and the sensors share a long-term secret key $SK_{GS} = h(ID_{Sn}||y)$. Shi et al.’s protocol consists of four phases; user registration phase, login phase, authentication phase, and password update phase. For convenience, the notations used throughout this paper are summarized in Table 2.

| Symbol | Description |
|--------|-------------|
| $p, q$ | Two large prime numbers |
| $F_P$ | A finite field |
| $E$ | An elliptic curve defined on finite field $F_P$ with large order |
| $G$ | The group of elliptic curve points on $E$ |
| $ID_U$ | The identity of user $U$ |
| $ID_{Sn}$ | The identity of sensor $S_n$ |
| $pw_U$ | The user $U$’s password |
| $GW$ | The gateway of WSN |
| $x, y$ | The master keys of $GW$ |
| $h(\cdot)$ | A secure one-way hash function |
| $\|$ | A string concatenation operation |
| $\oplus$ | A bitwise XOR operation |

3.1. Registration Phase

In this phase, the user $U$ securely submits its identity $ID_U$ and password $pw_U$ to the gateway $GW$. Then, $GW$ issues $U$ a smart card containing the user authentication information, as shown in Figure 1.

3.2. Login and Authentication Phases

In the login and authentication phases, when $U$ enters $ID_U$ and $pw_U$ into a smart card terminal, the smart card must validate the legitimacy of $U$. Then, $U, S_n$ and $GW$ authenticate with each other. This protocol uses 4 messages ($M_1, M_2, M_3, M_4$) for mutual authentication, as described in Figure 2. Lastly, $U$ and $S_n$ share the session key $sk$. After the authentication phase, $U$ and $S_n$ communicate with each other using the session key $sk$. 
**Figure 1.** The registration phase of Shi et al.’s protocol.

\[
\begin{align*}
\text{User (U)} & \quad \text{Gateway (GW)} \\
\text{chooses } ID_U, pw_U & \quad \text{computes } h(ID_U \oplus pw_U) \\
generates a random number } b_U & \quad \text{computes } X = h(ID_U \oplus X_U) \\
\text{computes } pw_U = h(pw_U \oplus b_U) & \quad \text{computes } K_U = h(ID_U \| x) \times P \\
\{ID_U, pw_U\} & \quad B_U = h(ID_U \oplus pw_U) \\
\text{inputs } b_U \text{ into the smart card } & \quad W_U = h(ID_U \| pw_U) \oplus K_U \\
\langle ID_U, pw_U \rangle & \quad \text{smart card } \langle B_U, W_U, h(\cdot) \rangle
\end{align*}
\]

**Figure 2.** The login and authentication phases of Shi et al.’s protocol.

\[
\begin{align*}
\text{User (U)} & \quad \text{Sensor (S_u)} & \quad \text{Gateway (GW)} \\
\text{computes } pw_U & \quad \text{sends } PW_U = h(ID_U \| \alpha) \\
\text{computes } Y_U = h(ID_U \| \alpha) & \quad \text{checks } T'' - T_U \leq \Delta T \\
\text{checks } X' = h(ID_U \| X) & \quad \text{generates } Y = r_S \times P \\
\text{computes } X_U = h(ID_U \| X) & \quad \beta = h(SK_{GS} \| ID_U \| X \| T_U \| \alpha \| ID_{S_u} \| Y \| T_S) \\
\text{checks } \gamma = h(SK_{GS} \| ID_U \| X \| T_U \| \alpha \| ID_{S_u} \| Y \| T_S \| T_G) & \quad \gamma = h(SK_{GS} \| ID_U \| X \| T_U \| \alpha \| ID_{S_u} \| Y \| T_S \| T_G) \\
\text{checks } \delta = h(ID_U \| X \| X' \| T_U \| Y \| T_S) & \quad \delta = h(ID_U \| X \| X' \| T_U \| Y \| T_S) \\
\text{checks } \tau = h(Y \| T_S \| \delta) \| K_{SU} & \quad \tau = h(Y \| T_S \| \delta) \| K_{SU} \\
\text{checks } sk = h(X \| Y \| K_{SU}) & \quad sk = h(X \| Y \| K_{SU})
\end{align*}
\]
3.3. Password Update Phase

In the password update phase, $U$ enters the identity $ID_U$, the old password $pw_U$, and the new password $pw'_U$. Then, the smart card updates the password after first checking the correctness of the old password, as shown in Figure 3.

**Figure 3.** The password update phase of Shi et al.’s protocol.

- User ($U$)
  - inserts smart card
  - enters $ID_U$, $pw_U$, $pw'_U$
  - $pw_U = h(pw_U \oplus b_U)$
  - $B'_U = h(ID_U \oplus pw_U)$
  - checks $B'_U = B_U$
  - $K_U = h(ID_U \| pw_U) \oplus W_U$
  - $pw'_U = h(pw'_U \oplus b_U)$
  - $W'_U = h(ID_U \| pw'_U) \oplus K_U$
  - $B_U = h(ID_U \oplus pw'_U)$
  - replaces $W_U$ with $W'_U$

4. Security Weaknesses in Shi et al.’s Protocol

This section shows that Shi et al.’s protocol is vulnerable to a session key attack, a stolen smart card attack, and a sensor energy exhausting attack.

4.1. Session Key Attack

In Shi et al.’s protocol, the user $U$ and the sensor $S_n$ have to perform the login and authentication phases when they want to share a session key which will be used for protecting their subsequent communication. A problem occurs if $U$ shares its session key with an attacker, not with the intended sensor $S_n$. In the protocol, the gateway $GW$ and the user $U$ check each other’s legitimacy using the authenticators $\alpha$ and $\delta$, respectively. However, $\alpha$ and $\delta$ do not include information about the sensor $S_n$ with which $U$ intends to establish a session key. The attacker exploits this design flaw in mounting a session key attack. The attack is depicted in Figure 4 and its description follows.

When $U$ inputs $ID_U$ and $pw_U$, and sends $M_1$ to sensor $S_n$, the attacker intercepts $M_1$ and sends it to sensor $S_A$ which was previously stolen by the attacker. Upon receiving $M_1$, the stolen sensor $S_A$ will generate the message $A_2$ and send it to the gateway $GW$. However, the attacker replaces $ID_{S_n}$ contained in $A_2$ with $ID_A$ to make $GW$ believe that $ID_U$ wants to communicate with sensor $S_A$, not with $S_n$. After receiving $A_2$, the gateway $GW$ generates $A_3$ without noticing any discrepancy and sends it to sensor $S_A$. Lastly, the attacker sends the user $U$ the message $A_4$ generated by $S_A$ using the message $A_3$ from $GW$. Because there is no information about the sensor $S_n$ in $A_4$ and $M_4$, the user $U$ undoubtedly shares the session key with the attacker while thinking that it has shared the key with the sensor $S_n$. 
Kocher et al. and Messerges et al. pointed out that the confidential information stored in smart cards could be extracted by physically monitoring its power consumption [13,14]. Therefore, it is fair to say that if a user loses his or her smart card, all information in the smart card may be revealed to the attacker.

In Shi et al.’s protocol, the smart card stores various information for user login and authentication. The smart card for the user $ID_U$ includes $b_U$, $B_U$, $W_U$ and $h(\cdot)$. Using these information and $ID_U$, an attacker can guess $U$’s password $pw_U$. If $ID_U$ is used in public communication, the attacker can obtain or steal it without difficulty. Figure 5 describes a stolen smart card attack against Shi et al.’s protocol.

The attacker can obtain information from the smart card using attacks such as simple power analysis (SPA) and differential power analysis (DPA). This information includes $b_U$, $B_U$, $W_U$ and $h(\cdot)$. Recall that $B_U = h(ID_U \oplus h(pw_U \oplus b_U))$. Using $B_U$ as a password verifier, the attacker can easily find out the password $pw_U$ by mounting an offline password guessing attack (also known as an offline dictionary attack) [19–22] if the password $pw_U$ is not long enough. After successfully mounting the password guessing attack, the attacker can login and authenticate with the sensor $S_n$ and the gateway $GW$ using the identity $ID_U$ and the password $pw_U$.

**Figure 4.** A session key attack on Shi et al.’s protocol.
Figure 5. A stolen smart card attack on Shi et al.'s protocol.

Attacker

gets $ID_U$ in public communication channel
gets(steals) user’s smart card
obtains information from smart card using SPA and DPA
→ gets $b_U, B_U, W_U$ and $h(\cdot)$

$B_U = h(ID_U \oplus pw_U)$
$\overline{pw}_U = h(pw_U \oplus b_U)$
→ $B_U = h(ID_U \oplus h(pw_U \oplus b_U))$

executes off-line password attack
→ figures out user’s password $pw_U$
→ logs in to WSNs using $ID_U$ and $pw_U$

4.3. Sensor Energy Exhausting Attack

The computational cost of a sensor is a critical consideration in the design of WSNs as it increases the consumption of the battery power of the sensor. Often it is economically advantageous to discard a sensor rather than recharge it. For this reason, the battery power of a sensor is usually important in wireless devices, with its lifetime determining the sensor lifetime. Previous work have suggested several types of energy exhausting attacks. Buttyan et al. [23] investigated the reliability of transport protocols for WSNs. Brownfield et al. [24] researched the battery depletion effect through the reduction of sleep cycles. Khouzani et al. [25,26] investigated malware attacks in battery-constrained wireless networks. As shown by the previous researches, WSNs need to eliminate unnecessary computational costs of sensors so that the effects of an energy exhausting attack on sensors can be minimized.

In Shi et al.’s protocol, the sensor performs various cryptographic operations such as one-way hash function evaluations, scalar-point multiplications, random number generations, and map-to-point hash function evaluations. Scalar-point multiplications are much more expensive than hash function evaluations. The computational costs of generating a random number and evaluating a map-to-point hash function are about half the cost of performing a scalar-point multiplication. A sensor consumes a large amount of energy to perform a scalar-point multiplication and very little to perform a hash function evaluation [27–29].

Figure 6 shows the possibility of a sensor energy exhaustion attack. The attacker can keep sending malicious messages, $A_1, A_2, A_3$, generated to consume the battery power of the sensor. The attacker can do so because the sensor only checks the freshness of the timestamp in $M_1$. For each of these fake messages, the sensor checks the freshness of the timestamp and proceeds to perform the subsequent cryptographic operations, thereby consuming large amounts of energy. Accordingly, it is necessary to modify the protocol so that the sensor can check if the message $M_1$ is from a legitimate user, not from an imposter.
Figure 6. A sensor energy exhausting attack on Shi et al.’s protocol.

| Attacker | Sensor $S_n$ |
|----------|-------------|
| $M_1 = \{ID_U, X, T_U, \alpha\} \rightarrow$ | checks $T' - T_U \leq \Delta T$ |
| $A_1 = \{*, *, T_U, *\} \rightarrow$ | generates $r_S \in \mathbb{Z}_q^*$ |
| $A_2 = \{*, *, T_U, *\} \rightarrow$ | $Y = r_S \times P$ |
| $A_3 = \{*, *, T_U, *\} \rightarrow$ | $\beta = h(SK_{GS} || ID_U || X || T_U || \alpha || ID_{S_n} || Y || T_S)$ |
| $\vdots$ | $M_2 = \{ID_U, X, T_U, \alpha, ID_{S_n}, Y, T_S, \beta\} \rightarrow$ |

After checking only the timestamp $T_U$, the sensor:

- generates a random number
- performs a scalar-point multiplication
- performs a hash function evaluation

5. The Proposed Protocol

Like Shi et al.’s protocol, our proposed protocol is divided into three phases: the user registration phase, login and authentication phase, and password update phase. Before the protocol is ever executed, the gateway generates two master keys, $x$ and $y$, and shares a long-term secret key $SK_{GS} = h(ID_{S_n} || y)$ with the sensor $S_n$. In describing the protocol, we use the same notations as in Table 2 unless stated otherwise.

5.1. Registration Phase

For a user $U$, this phase is performed only once when $U$ registers itself with the gateway $GW$. Figure 7 illustrates how the phase works, and its description follows:

Figure 7. The registration phase.

(1) The user $U$ chooses its identity $ID_U$ and password $pw_U$ freely, generates a random number $b_U$, and computes $\overline{pw}_U = h(pw_U \oplus b_U)$. $U$ sends $ID_U$ and $\overline{pw}_U$ to $GW$ via a secure channel.

(2) The gateway $GW$ computes:
\[ K_U = h(ID_U \| x) \times P \]
\[ A_U = \overline{pw}_U \oplus h(x \oplus y) \]
\[ B_U = h(ID_U \| \overline{pw}_U \| h(x \oplus y)) \]
\[ W_U = h(ID_U \| \overline{pw}_U) \oplus K_U \]

Then, GW issues U a smart card loaded with \( \{ A_U, B_U, W_U, h(\cdot) \} \).

(3) Lastly, U inputs the random number \( b_U \) into the smart card.

5.2. Login and Authentication Phase

This phase is carried out whenever U wants to gain access to the WSN. During the phase, U establishes a session key with the sensor \( S_n \) while being authenticated by the gateway GW. The phase proceeds as follows (see also Figure 8):

**Step 1.** U inserts its smart card into the card reader and inputs its identity \( ID_U \) and password \( pw_U \).

Then, the smart card computes:
\[ \overline{pw}_U = h(pw_U \oplus b_U) \]
\[ B'_U = h(ID_U \| \overline{pw}_U \| h(x \oplus y)) \]

and checks if \( B_U \) is equal to \( B'_U \). If not equal, the smart card aborts the protocol. Otherwise, it retrieves the current timestamp \( T_U \), chooses a random number \( r_U \in \mathbb{Z}_q^* \), and computes:
\[ K_U = h(ID_U \| \overline{pw}_U) \oplus W_U \]
\[ X = r_U \times P \]
\[ X' = r_U \times K_U \]
\[ \omega = h(ID_U \| h(ID_{S_n} \| h(x \oplus y)) \| T_U) \]
\[ \alpha = h(ID_U \| ID_{S_n} \| X \| X' \| T_U \| \omega) \]

After the computations, the smart card sends the message \( M_1 = \langle ID_U, ID_{S_n}, X, T_U, \alpha, \omega \rangle \) to the sensor \( S_n \).

**Step 2.** Upon receiving \( M_1 \) from U, the sensor \( S_n \) retrieves the current timestamp \( T'_U \) and verifies the freshness of U’s timestamp \( T_U \) by checking that:
\[ T' - T_U \leq \Delta T \]

where \( \Delta T \) is the maximum allowed time difference between \( T_U \) and \( T'_U \). If \( T_U \) is not fresh, \( S_n \) rejects U’s request and aborts the protocol. Otherwise, \( S_n \) checks if \( \omega \) is equal to the hash value \( h(ID_U \| h(ID_{S_n} \| h(x \oplus y)) \| T_U) \). If they are not equal, \( S_n \) aborts the protocol. Otherwise, \( S_n \) generates a random number \( r_S \in \mathbb{Z}_q^* \), retrieves the current timestamp \( T_S \), and computes:
\[ Y = r_S \times P \]
\[ \beta = h(SK_{GS} \| ID_U \| X \| T_U \| \alpha \| \omega \| ID_{S_n} \| Y \| T_S) \]

Then, \( S_n \) sends the message \( M_2 = \langle ID_U, X, T_U, \alpha, \omega, ID_{S_n}, Y, T_S, \beta \rangle \) to the gateway GW.
Figure 8. The login and authentication phase.

User (U)  Sensor (Sn)  Gateway (GW)

\[
\begin{align*}
\overline{pw}_U &= h(pw_U \oplus b_U) \\
h(x \oplus y) &= \overline{pw}_U \oplus A_U \\
B'_U &= h(ID_U \| \overline{pw}_U \| h(x \oplus y)) \\
\text{checks} B'_U &= B_U \\
K_U &= h(ID_U \| \overline{pw}_U) \oplus W_U \\
\text{generates} r_U &\in \mathbb{Z}_q^* \\
&\text{retrieves } U'' \text{'s timestamp } T_U \\
X &= r_U \times P \\
X' &= r'_U \times K_U \\
\omega &= h(ID_U\|h(ID_{Sn}\|h(x \oplus y))\|T_U) \\
\alpha &= h(ID_U\|ID_{Sn}\|X\|X'\|T_U\|\omega) \\
\end{align*}
\]

\[ M_1 = \langle ID_U, ID_{Sn}, X, T_U, \alpha, \omega \rangle \]

\[
\begin{align*}
&\text{checks } T' - T_U \leq \Delta T \\
&\text{checks } \omega = h(ID_U\|h(ID_{Sn}\|h(x \oplus y))\|T_U) \\
&\text{generates} r_S &\in \mathbb{Z}_q^* \\
&\text{retrieves } S_{n}' \text{'s timestamp } T_S \\
Y &= r_S \times P \\
\beta &= h(SK_{GS}\|ID_U\|X\|T_U\|\alpha\|\omega\|ID_{Sn}\|Y\|T_S) \\
\end{align*}
\]

\[ M_2 = \langle ID_U, X, T_U, \alpha, \omega, ID_{Sn}, Y, T_S, \beta \rangle \]

\[
\begin{align*}
&\text{checks } T'' - T_S \leq \Delta T \\
&\text{checks } \beta = h(SK_{GS}\|ID_U\|X\|T_U\|\alpha\|\omega\|ID_{Sn}\|Y\|T_S) \\
&\text{retrieves } G W'' \text{'s timestamp } T_G \\
X' &= h(ID_U\|x\) \times X \\
\gamma &= h(SK_{GS}\|ID_U\|X\|T_U\|\alpha\|ID_{Sn}\|Y\|T_S\|T_G) \\
\delta &= h(ID_U\|X\|X'\|T_U\|ID_{Sn}\|Y\|T_S) \\
\end{align*}
\]

\[ M_3 = \langle T_G, \gamma, \delta \rangle \]

\[
\begin{align*}
&\text{checks } T''' - T_G \leq \Delta T \\
&\text{checks } \gamma = h(SK_{GS}\|ID_U\|X\|T_U\|\alpha\|ID_{Sn}\|Y\|T_S\|T_G) \\
&K_{SU} = r_S \times X \\
\text{retrieves } r_S' \text{'s new timestamp } T_S' \\
\tau &= h(Y\|T_S'\|\delta\|K_{SU}) \\
sk &= h(X\|Y\|K_{SU}) \\
\end{align*}
\]

\[ M_4 = \langle Y, T_S, T_S', \delta, \tau \rangle \]

\[
\begin{align*}
&\text{checks } T''' - T_S' \leq \Delta T \\
&\text{checks } \delta = h(ID_U\|X\|X'\|T_U\|ID_{Sn}\|Y\|T_S) \\
&K_{US} = r_U \times Y \\
\text{checks } \tau = h(Y\|T_S'\|\delta\|K_{US}) \\
sk &= h(X\|Y\|K_{US}) \\
\end{align*}
\]
Step 3. After receiving $M_2$, $GW$ retrieves the current timestamp $T''$ and verifies the freshness of the timestamp $T_S$ by checking that $T'' - T_S \leq \Delta T$. If $T_S$ is not fresh, $GW$ aborts the protocol. Otherwise, $GW$ computes $X' = h(ID_U || x) \times X$ and checks if $\alpha$ equals $h(ID_U || ID_{S_n} || X || X' || T_U || T_S || \omega)$ and $\beta$ equals $h(SK_{GS} || ID_U || X || T_U || T_S || ID_{S_n} || Y || T_S)$. If either of the checks fails, $GW$ aborts the protocol. Otherwise, $GW$ retrieves the current timestamp $T_G$ and computes:

$$\gamma = h(SK_{GS} || ID_U || X || T_U || \alpha || ID_{S_n} || Y || T_S || T_G)$$

$$\delta = h(ID_U || X || X' || T_U || ID_{S_n} || Y || T_S)$$

Then, $GW$ sends $M_3 = \langle T_G, \gamma, \delta \rangle$ to the sensor $S_n$.

Step 4. Having received $M_3$, $S_n$ retrieves the current timestamp $T''$ and checks if $T'' - T_G \leq \Delta T$ and $\gamma = h(SK_{GS} || ID_U || X || T_U || \alpha || ID_{S_n} || Y || T_S || T_G)$. Only if both the checks hold, $S_n$ retrieves the new timestamp $T'_S$ and computes:

$$K_{SU} = r_S \times X$$

$$\tau = h(Y || T'_S || \delta || K_{SU})$$

$$sk = h(X || Y || K_{SU})$$

Then, $S_n$ sends $M_4 = \langle Y, T_S, T'_S, \delta, \tau \rangle$ to the user $U$.

Step 5. With $M_4$ in hand, $U$ retrieves the current timestamp $T'''$, computes $K_{US} = r_U \times Y$, and checks if (1) $T''' - T'_S \leq \Delta T$; (2) $\delta = h(ID_U || X || X' || T_U || ID_{S_n} || Y || T_S)$; and (3) $\tau = h(Y || T'_S || \delta || K_{US})$. If any of the checks fail, $U$ aborts the protocol. Otherwise, $U$ computes:

$$sk = h(X || Y || K_{US})$$

5.3. Password Update Phase

Our protocol allows users to freely update their passwords. The password update phase works as follows (see also Figure 9):

1. The user $U$ inserts its smart card into a smart card reader and enters the identity $ID_U$, the old password $pw_U$, and the new password $pw'_U$.

2. The smart card computes $\overline{pw}_U = h(pw_U \oplus b_U)$, $h(x \oplus y) = A_U = \overline{pw}_U$, and $B'_U = h(ID_U || \overline{pw}_U || h(x \oplus y))$ and checks if $B'$ is equal to $B$. If they are not the same, the password update phase stops. Otherwise, the smart card computes:

$$K_U = W_U \oplus h(ID_U || \overline{pw}_U)$$

$$\overline{pw}'_U = h(pw'_U \oplus b_U)$$

$$A'_U = \overline{pw}'_U \oplus h(x \oplus y)$$

$$B'_U = h(ID_U || \overline{pw}'_U || h(x \oplus y))$$

$$W'_U = h(ID_U || \overline{pw}'_U) \oplus K_U$$

and replaces $A_U$, $B_U$ and $W_U$ with $A'_U$, $B'_U$ and $W'_U$, respectively.
Figure 9. The password update phase.

\[
\begin{align*}
\text{User (U)} & \quad \text{inserts smart card} \\
& \quad \text{enters } ID_U, pw_U, pw'_U \\
\overline{pw}_U &= h(pw_U \oplus b_U) \\
B'_U &= h(ID_U | \overline{pw}_U | h(x \oplus y)) \\
& \quad \text{checks } B'_U = B_U \\
K_U &= h(ID_U | \overline{pw}_U) \oplus W_U \\
h(x \oplus y) &= A_U \oplus \overline{pw}_U \\
\overline{pw'}_U &= h(pw'_U \oplus b_U) \\
A'_U &= \overline{pw'}_U \oplus h(x \oplus y) \\
B'_U &= h(ID_U | \overline{pw'}_U | h(x \oplus y)) \\
W'_U &= h(ID_U | \overline{pw'}_U) \oplus K_U \\
& \quad \text{replaces } A_U, B_U, W_U \text{ with } A'_U, B'_U, W'_U
\end{align*}
\]

6. Performance Comparison

Table 3 compares our improved protocol with Yeh et al.’s protocol [2] and Shi et al.’s protocol [3] in terms of the computational costs required by the protocols. The efficiency comparison is based on theoretical analysis and experimental results [3,27–29].

| Protocol          | Computational Cost |
|-------------------|--------------------|
|                   | User   | Sensor | Gateway                  |
| Yeh et al.’s protocol | $2M + 1R + 1A + 4H$ | $2M + 1R + 1A + 1P + 1H$ | $3M + 1R + 1P + 1H$ |
| Shi et al.’s protocol  | $3M + 5H$     | $2M + 3H$ | $1M + 4H$ |
| Our protocol       | $3M + 7H$     | $2M + 4H$ | $1M + 4H$ |

Notations used in Table 3 are described as follows:

- $M$: scalar-point multiplication
- $R$: random point generation
- $A$: point addition
- $P$: map-to-point hash function evaluation
- $H$: hash function evaluation

The computational costs of generating a random point and evaluating a map-to-point hash function are about half the cost of performing a scalar-point multiplication. Hash function evaluations and point addition operations are often ignored in cost estimates since they are much faster than scalar-point multiplications. If we ignore hash function evaluations, the computational costs described in Table 3 can be estimated as in Table 4.
Table 4. Estimated efficiency comparison.

| Protocol                | User | Sensor | Gateway |
|-------------------------|------|--------|---------|
| Yeh et al.’s protocol   | 2.5M | 3M     | 3M      |
| Shi et al.’s protocol   | 3M   | 2M     | 1M      |
| Our protocol            | 3M   | 2M     | 1M      |

As shown in Tables 3 and 4, our proposed protocol and Shi et al.’s protocol are more efficient than Yeh et al.’s protocol, in terms of the computational costs of the sensor and the gateway. In WSNs, it is important to minimize the energy consumption of the sensor node. In this sense, it is fair to say that our protocol and Shi et al.’s protocol are better suited for WSNs than Yeh et al.’s protocol. The performance of our proposed protocol is similar to that of Shi et al.’s protocol. But, as we demonstrated in Section 4, Shi et al.’s protocol is vulnerable to a session key attack, a stolen smart card attack, and a sensor energy exhausting attack. Consequently, we can say that our protocol enhances the security of Shi et al.’s protocol while maintaining the efficiency of the protocol.

7. Security Analysis and Verification

In this section, we first provide a heuristic security analysis for the proposed protocol and then formally verify the security analysis by using Rubin logic.

7.1. Heuristic Security Analysis

7.1.1. Stolen-Verifier Attack

In WSNs, an attacker may attempt to mount a stolen-verifier attack if the gateway stores a password verifier [30] and then, impersonate a legal user using the verifier stolen from the gateway. However, in our protocol, the gateway does not store a password verifier of any kind but stores only the master secret keys $x$ and $y$ which are used in computing:

$$SK_{GS} = h(ID_S || y)$$

$$X' = h(ID_U || x) \times X$$

7.1.2. Insider Attack

An insider attack occurs when the gateway manager or system administrator can access a user’s secret (e.g., user password) and then impersonate the user. However, in our protocol, the user $U$ does not send a plain password to the gateway, but sends only the password-derived hash value $pw_U = h(pw_U \oplus b_U)$. Since $b_U$ is a sufficiently high-entropy random number, the gateway cannot learn the password $pw_U$ from the hash value $pw_U$. In addition, the gateway does not manage any table for storing user passwords or their verifiers (e.g., an ID/password table) Therefore, an insider attack is not possible against our protocol.
7.1.3. Replay Attack

In our protocol, each of the protocol messages \((M_1, M_2, M_3 \text{ and } M_4)\) accompanies at least one of the authenticators \((\alpha, \beta, \gamma, \delta, \tau \text{ and } \omega)\) which are generated using a timestamp \((T_U, T_S, T'_S \text{ or } T_G)\) as part of the hash input. The protocol participants \((U, S_n \text{ and } GW)\) verify the authenticity of incoming messages by checking the freshness of the timestamps and the legitimacy of the authenticators. But, an attacker cannot compute any of the authenticators for a fresh timestamp without knowing an appropriate secret. Therefore, our proposed protocol is secure against replay attacks.

7.1.4. Man-in-the-Middle Attack

It is impossible for an attacker to mount a man-in-the-middle attack against our proposed protocol. In a typical man-in-the-middle attack, an attacker intercepts the messages being exchanged between the communicating parties and instead, sends arbitrary messages for its own benefit impersonating one of them to the other. But, our protocol allow the parties to authenticate all the protocol messages with the authenticators \(\alpha, \beta, \gamma, \delta, \tau \text{ and } \omega\), and therefore, is secure against man-in-the-middle attacks.

7.1.5. Gateway Impersonation Attack

An attacker cannot impersonate the gateway because it cannot forge the message:

\[
M_3 = \langle T_G, \gamma, \delta \rangle
\]

To generate \(\gamma\) or \(\delta\), one needs to know either \(SK_{GS}\) or \(h(ID_U || x)\). However, \(h(ID_U || x)\) is the secret shared only between the user and the gateway while \(SK_{GS}\) is the secret shared between the sensor and the gateway. Therefore, it is impossible for an attacker to mount a gateway impersonation attack.

7.1.6. User Impersonation Attack

It is impossible for an attacker to impersonate the user as it cannot forge the message:

\[
M_1 = \langle ID_U, ID_{S_n}, X, T_U, \alpha, \omega \rangle
\]

The attacker should know \(X'\) to compute \(\alpha\) and should know \(h(x \oplus y)\) to compute \(\omega\). But, the attacker knows neither \(X'\) nor \(h(x \oplus y)\) and therefore, cannot mount a user impersonation attack.

7.1.7. Sensor Impersonation Attack

An attacker cannot impersonate the sensor because it can forge the messages \(M_2 = \langle ID_U, X, T_U, \alpha, \omega, ID_{S_n}, Y, T_S, \beta \rangle\) and \(M_4 = \langle Y, T_S, T'_S, \delta, \tau \rangle\). The attacker cannot compute \(\beta\) without knowing \(SK_{GS}\) and cannot compute \(\delta\) without knowing the secret \(h(ID_U || x)\). But, the attacker knows neither \(SK_{GS}\) nor \(x\) and therefore, cannot mount a sensor impersonation attack.
7.1.8. Mutual Authentication

Mutual authentication is an important security property that an authentication protocol should achieve [31,32]. Our proposed protocol provides mutual authentication among the three parties: the user, the sensor and the gateway.

- The gateway authenticates the user using $\alpha$ in $M_2$.
- The gateway authenticates the sensor using $\beta$ in $M_2$.
- The sensor authenticates the gateway using $\gamma$ in $M_3$.
- The user authenticates the gateway using $\delta$ in $M_4$.
- The user and the sensor authenticate each other via $\delta$ from the gateway.

This means that our protocol achieves mutual authentication.

7.1.9. Perfect Forward Secrecy

Perfect forward secrecy means that a session key derived from a set of long-term keys will not be compromised even if one of the long-term keys is compromised in the future. The proposed protocol uses the session key $sk = h(X\parallel Y\parallel r_S \times X)$ for the sensor and $sk = h(X\parallel Y\parallel r_U \times Y)$ for the user. Even though $h(ID_U\parallel x)$ and $x$ are compromised, an attacker cannot know $r_U$ or $r_S$. Under the assumption that the ECDHHP problem is hard, the attacker cannot compute $r_S$ from $r_S \times X$ and $r_U$ from $r_U \times Y$. Therefore, our protocol provides perfect forward secrecy.

7.1.10. Key Agreement

The proposed protocol provides key agreement between the user and the sensor. To the session-key computation, the user contributes its random number $r_U$ while the sensor contributes its random number $r_S$. It is straightforward to verify that $K_{SU}$ and $K_{US}$ are equal:

$$K_{SU} = r_S \times X = r_S \times r_U \times P$$
$$K_{US} = r_U \times Y = r_U \times r_S \times P$$

Since $K_{SU} = K_{US}$, it is clear that the user and the sensor compute session keys of the same value:

$$sk = h(X\parallel Y\parallel K_{US})$$
$$= h(X\parallel Y\parallel K_{SU})$$

7.1.11. Session Key Attack

In our protocol:

- $\alpha$ is combined with two identities $ID_U$ and $ID_{S_n}$, which indicates that the user $U$ wants to communicate with the sensor $S_n$.
- $\delta$ is also combined with $ID_U$ and $ID_{S_n}$, which indicates that the gateway has authenticated both the user $ID_U$ and the sensor $ID_{S_n}$.

But, no attacker can compute $\alpha$ and $\delta$, and therefore, can share a session key with the user.
7.1.12. Stolen Smart Card Attack

In Shi et al.’s protocol, the attacker can obtain $b_U$ and $B_U$ from the smart card and thus can use $B_U = h(ID_U \oplus h(pw_U \oplus b_U))$ as the password verifier in its offline dictionary attack. However, in our protocol, $B_U$ is computed as $B_U = h(ID_U \| pw_U \| h(x \oplus y))$. Even if the attacker obtains $b_U$ and $B_U$ from the smart card, it cannot use $B_U$ as a password verifier since it does not know the hash value $h(x \oplus y)$. Therefore, no attacker can mount an offline dictionary attack against our protocol.

7.1.13. Sensor Energy Exhausting Attack

In Shi et al.’s protocol, the sensor has to generate a random number and execute a scalar-point multiplication whenever it receives the message $M_1$ from the user. Random number generations and scalar-point multiplications are expensive and exhaust a large amount of the sensor’s energy. This makes Shi et al.’s protocol vulnerable to a sensor energy exhausting attack. However, in our protocol, the sensor first checks the validity of $\omega = h(ID_U \| h(ID_{Sn} \| h(x \oplus y)) \| T_U)$ before generating a random number and performing a scalar-point multiplication. Checking the validity of $\omega$ only requires one hash function evaluation. Therefore, our proposed protocol is secure against a sensor energy exhausting attack.

Table 5 summarizes and compares the security of our protocol, Yeh et al.’s protocol, and Shi et al.’s protocol.

| Attack and Security Property | Yeh et al.’s Protocol | Shi et al.’s Protocol | Our Protocol |
|------------------------------|-----------------------|-----------------------|-------------|
| Stolen-verifier attack       | Secure                | Secure                | Secure      |
| Insider attack               | Secure                | Secure                | Secure      |
| Replay attack                | Secure                | Secure                | Secure      |
| Man-in-the-middle attack     | Secure                | Secure                | Secure      |
| Gateway impersonation attack | Secure                | Secure                | Secure      |
| User impersonation attack    | Secure                | Secure                | Secure      |
| Sensor impersonation attack  | Insecure              | Secure                | Secure      |
| Mutual authentication        | No                    | Yes                   | Yes         |
| Perfect forward secrecy      | No                    | Yes                   | Yes         |
| Key agreement between user and sensor | No | Yes | Yes |
| Session key attack           | Insecure              | Insecure              | Secure      |
| Stolen smart card attack     | Insecure              | Insecure              | Secure      |
| Sensor energy exhausting att | Insecure              | Insecure              | Secure      |

7.2. Rubin Logic Verification

We analyze the proposed protocol using Rubin logic which can be applicable in analyzing an authentication protocol. Rubin logic integrates protocol analysis with specification and uses the notions of global sets, local sets, and actions. As the protocol run is progressed, the possession and belief sets (specified by local sets) are modified for each principal by inference rules (specified by global sets) and
actions [33,34]. As the possession and belief sets are modified, secret set and observers sets (specified by global sets) are modified as well.

**Global Sets.** The first step of the specification of any protocol using Rubin logic is to instantiate the global sets with values. Global sets are public to each principal in a protocol specification.

- Principal Set: This set contains the principals who participate in a protocol.
- Rule Set: This set contains inference rules for deriving new statements from existing assertions.
- Secret Set: This set contains all of the secrets that exist at any given time in the system.
- Observers Sets: For each secret, its set contains all the principals who could possibly know the secret by listening to network traffic or generating it themselves.

**Local Sets.** Local sets are private to each principal in a protocol specification [35]. For each principal, $P_i$, Rubin logic defines the following sets:

- Possession Set($P_i$): This set contains all the data relevant to security that this principal knows or possesses. We denote this set by $\text{POSS}(P_i) = (\text{poss}_{1}, \text{poss}_{2}, \ldots, \text{poss}_{n})$.
- Belief Set($P_i$): This set contains all the beliefs hold by a principal. For example, the keys it holds between itself and other principals, beliefs about jurisdiction, beliefs about freshness, and beliefs about the possessions of other principals. We denote this set by $\text{BEL}(P_i) = (\text{bel}_{1}, \text{bel}_{2}, \ldots, \text{bel}_{n})$.
- Behavior List($P_i$): This item is a list rather than a set because the elements are ordered. $\text{BL}(P_i) = \text{Behavior List of } P_i$.

**Actions.** Rubin logic defines actions for dealing with the knowledge in a protocol [36]. The action lists that precede and follow message operations in a principal’s behavior list determine a sequence of events performed by the principal during a protocol run. We use the following actions:

- Generate-nonce($N$)
- Send($P_i, X$)
- Receive($P_i, X$)
- Update($X$)
- Forget($X$)
- Concat($X_1, X_2, \ldots, X_n$)
- XOR($X_1, X_2, \ldots, X_n$)
- Check($X_1, X_2, \ldots, X_n$)
- Scalar-multiplication($X_1, X_2, \ldots, X_n$)
- Hash($h(\cdot); X_1, X_2, \ldots, X_n$)
- Check-freshness($T$)

Here, Concat($X_1, X_2, \ldots, X_n$) is the action that concatenates the submessages $X_1, X_2, \ldots, X_n$.

### 7.2.1. Protocol Specification

Notations used for the protocol specification is the same as those in Table 2. Phases 1, 2 and 3 represent the registration phase, the login and authentication phase, and the password updated phase. The global and local sets for the protocol are specified as follows:
Global Sets. The global sets are specified as follows:

- Principal set: A principal is one of \( U \), \( S_n \) and \( GW \). \( U \) is the protocol initiator.
- Rule set:
  - \( X \) contains \( Y \): \( Y \) appears as a submessage of \( X \).
  - \( S := f(S) \): \( S \) is replaced by the value \( f(S) \).
  - \( X \) from \( E \): \( X \) is received from \( E \).
  - \( \text{LINK}(N) \): \( \text{LINK} \) is used to link responses to challenges. When a principal generates a nonce, \( N \), the formula \( \text{LINK}(N) \) is added to the belief set of the principal.
- Secret Set: \( \{ pw_U, b_U, x, y, h(x \oplus y), SK_{GS} \} \)
- Observers Sets:
  - Observers\((pw_U)\) : \( \{ U \} \)
  - Observers\((b_U)\) : \( \{ U \} \)
  - Observers\((x)\) : \( \{ GW \} \)
  - Observers\((y)\) : \( \{ GW \} \)
  - Observers\((h(x \oplus y))\) : \( \{ S_n, GW \} \)
  - Observers\((SK_{GS})\) : \( \{ S_n, GW \} \)

Local Sets. The local sets are defined for each \( U \), \( S_n \) and \( GW \). Tables 6–8 show the specification of the local sets for \( U \), \( S_n \) and \( GW \), respectively.

Table 6. Local sets specification for principal \( U \).

| Principal U |
|-------------|
| \( \text{POSS}(U) = \{ pw_U, b_U, \{ ID_U \} \} \) |
| \( \text{BEL}(U) = \{ \emptyset(pw_U), \emptyset(b_U) \} \) |
| \( \text{BL}(U) \) |
| Phase 1 |
| (U1) \( \overline{pw}_U \leftarrow \text{Hash}(h(\cdot) \oplus \text{XOR}(pw_U, b_U)) \) |
| (U2) Send\((GW, \{ ID_U, \overline{pw}_U \}) \) |
| (U3) Update\((ID_U, \overline{pw}_U) \) |
| (U4) Receive\((GW, \{ A_U, B_U, W_U, h(\cdot) \}) \) |
| Phase 2 |
| (U5) \( \overline{pw}_U \leftarrow \text{Hash}(h(\cdot) \oplus \text{XOR}(pw_U, b_U)) \) |
| (U6) \( h(x \oplus y) \leftarrow \text{XOR}(\overline{pw}_U, A_U) \) |
| (U7) \( B'_U \leftarrow \text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \overline{pw}_U, h(x \oplus y))) \) |
| (U8) Check\((B'_U, B_U) \) |
| (U9) \( K_U \leftarrow \text{XOR}(\text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \overline{pw}_U, W_U))) \) |
| (U10) Generate-nonce\((r_U) \) |
| (U11) \( X \leftarrow \text{Scalar-multiplication}(r_U, P) \) |
| (U12) \( X' \leftarrow \text{Scalar-multiplication}(r_U, K_U) \) |
| (U13) \( \omega \leftarrow \text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \text{Hash}(h(\cdot) \oplus \text{ID}_{S_n}, h(x \oplus y)), T_U)) \) |
| (U14) \( \alpha \leftarrow \text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \text{ID}_{S_n}, X, X', T_U, \omega)) \) |
| (U15) Send\((S_n, \{ ID_U, ID_{S_n}, X, T_U, \omega, \alpha \}) \) |
| (U16) Update\((ID_U, ID_{S_n}, X, T_U, \alpha, \omega) \) |
| (U17) Receive\((S_n, \{ Y, T_S, \delta, \gamma \}) \) |
| (U18) Check-freshness\((T'_S) \) |
| (U19) Check \( \) |
| (U20) \( K_{US} \leftarrow \text{Scalar-multiplication}(r_U, Y) \) |
| (U21) Check\((\tau, \text{Hash}(h(\cdot) \oplus \text{Concat}(Y, T'_S, \delta, K_{US}))) \) |
| (U22) \( sk \leftarrow \text{Hash}(h(\cdot) \oplus \text{Concat}(X, Y, K_{US})) \) |
| Phase 3 |
| (U23) \( \overline{pw}_U \leftarrow \text{Hash}(h(\cdot) \oplus \text{XOR}(pw_U, b_U)) \) |
| (U24) \( B'_U \leftarrow \text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \overline{pw}_U, h(x \oplus y))) \) |
| (U25) Check\((B'_U, B_U) \) |
| (U26) \( K_{US} \leftarrow \text{XOR}(\text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \overline{pw}_U)), W_U) \) |
| (U27) \( h(x \oplus y) \leftarrow \text{XOR}(\overline{pw}_U, A_U) \) |
| (U28) \( \overline{pw}_U \leftarrow \text{Hash}(h(\cdot) \oplus \text{XOR}(pw'_U, A_U)) \) |
| (U29) \( A'_U \leftarrow \text{XOR}(\overline{pw}_U, h(x \oplus y)) \) |
| (U30) \( B'_U \leftarrow \text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \overline{pw}_U, h(x \oplus y))) \) |
| (U31) \( W'_U \leftarrow \text{XOR}(\text{Hash}(h(\cdot) \oplus \text{Concat}(ID_U, \overline{pw}_U)), K_U) \) |
| (U32) \( A_U \leftarrow A'_U \) |
| (U33) \( B_U \leftarrow B'_U \) |
| (U34) \( W_U \leftarrow W'_U \) |
Table 7. Local sets specification for principal $S_n$.

| Principal $S_n$                      |
|--------------------------------------|
| POSS$(S_n) = \{ SK_GS, h(x@y), \{ ID_{S_n} \} \}$                             |
| BEL$(S_n) = \{ \exists (SK_GS), \exists (h(x@y)) \}$                          |
| BL$(S_n)$                                                                 |
| Phase 2                                                                           |
| (SN1) Receive$(U, \{ I_D, I_{D_{S_n}}, X, T_U, \alpha, \omega \})$             |
| (SN2) Check-freshness$(T_U)$                                                     |
| (SN3) Check                                                                      |
| $(\omega, Hash(h(\cdot); Concat(I_D, Hash(h(\cdot); I_{D_{S_n}}, h(x@y)), T_U)))$ |
| (SN4) Generate-nonce$(r_S)$                                                      |
| (SN5) $Y \leftarrow$ Scalar-multiplication$(r_S, P)$                           |
| (SN6) $\beta \leftarrow$ Hash                                                  |
| $(h(\cdot); Concat(SK_GS, I_D, X, T_U, \alpha, \omega, ID_{S_n}, Y, T_S))$      |

Table 8. Local sets specification for principal $GW$.

| Principal $GW$                  |
|---------------------------------|
| POSS$(GW) = \{ x, y, h(x@y), SK_GS \}$                                      |
| BEL$(GW) = \{ \exists (x), \exists (y), \exists (h(x@y)), \exists (SK_GS) \}$ |
| BL$(GW)$                                                                      |
| Phase 1                                                                       |
| (GW1) Received$(U, \{ I_D, \overline{pw}_U \})$                             |
| (GW2) $K_U \leftarrow$                                                       |
| Scalar-multiplication$(Hash(h(\cdot); Concat(I_D, x), P)$                     |
| (GW3) $A_U \leftarrow$ XOR$(\overline{pw}_U, h(x@y))$                        |
| (GW4) $B_U \leftarrow$ Hash$(h(\cdot); Concat(I_D, \overline{pw}_U, h(x@y)))$ |
| (GW5) $W_U \leftarrow$ XOR$(Hash(h(\cdot); Concat(I_D, \overline{pw}_U)), K_U)$ |
| (GW6) Send$(U, \{ A_U, B_U, W_U, h(\cdot) \})$                              |
| (GW7) Update$(A_U, B_U, W_U, h(\cdot))$                                      |
| (GW8) Forget$(I_D, \overline{pw}_U, A_U, B_U, K_U, W_U)$                     |
| (GW9) Receive$(S_n, \{ I_D, X, T_U, \alpha, \omega, ID_{S_n}, Y, T_S, \beta \})$ |
| (GW10) Check-freshness$(T_S)$                                                 |
| (GW11) Check                                                                  |
| $(\beta, Hash(h(\cdot); Concat(SK_GS, I_D, X, T_U, \alpha, ID_{S_n}, Y, T_S)))$ |
| (GW12) $X' \leftarrow$ Scalar-multiplication$(Hash(h(\cdot); Concat(I_D, x), X)$ |
| (GW13) $Check(\alpha, Hash(h(\cdot); Concat(I_D, I_{D_{S_n}}, X, X', T_U, \omega))$ |
| (GW14) $\gamma \leftarrow$ Hash$(h(\cdot); Concat(SK_GS, I_D, X, T_U, \alpha, ID_{S_n}, Y, T_S, T_G))$ |
| (GW15) $\delta \leftarrow$ Hash$(h(\cdot); Concat(I_D, X, X', T_U, I_{D_{S_n}}, Y, T_S))$ |
| (GW16) Send$(S_n, \{ T_G, \gamma, \delta \})$                                |
| (GW17) Update$(T_G, \gamma, \delta)$                                         |

7.2.2. Analysis and Verification

In phase 1, $U$ initiates the protocol, and then the actions in BL$(U)$ are performed. Firstly, (U1)–(U3) actions in BL$(U)$ are performed, which represent that $U$ sends $I_D$ and $\overline{pw}_U$ to $GW$ for registration. Next, (GW1)–(GW8) actions in BL$(GW)$ are performed to generate $A_U, B_U, K_U$ and $W_U$, and to send them to $U$. By (GW8), $GW$ deletes $I_D, \overline{pw}_U, A_U, B_U, K_U$ and $W_U$ from POSS$(GW)$ and BEL$(GW)$. Lastly, the (U4) action in BL$(U)$ is executed, then phase 1 is finished. Due to the (GW8) forget action, the local sets of $GW$ are not changed. However, the local sets of principal $U$ are changed as described below.

- POSS$(U) = \{ pw_U, b_U, \overline{pw}_U, \{ I_D \}, \{ A_U, B_U, W_U, h(\cdot) \} \}$ from $GW$
- BEL$(U) = \{ \exists (pw_U), \exists (b_U), \exists (\overline{pw}_U) \}$

Accordingly, the global sets are modified as follows:
• Secret set: \(\{pw_U, b_U, \overline{pw}_U, x, y, SK_{GS}, h(x \oplus y)\}\)
• Observers sets:
  – Observers(\(\overline{pw}_U\)): \(\{U\}\)

In the (U5)–(U8) actions in BL(U) of phase 2, the smart card authenticates U, who inputs \(ID_U\) and \(pw_U\), by checking whether \(B_U\) and \(B'_U\) are same or not. Next, the (U9)–(U15) actions are executed to generate the protocol values \(X, X', h(x \oplus y), \omega, \alpha\) and \(r_U\). After the (U16) update action, the local sets of U are changed as follows:

• \(\text{POSS}(U) = \{ID_{S_n}, pw_U, b_U, \overline{pw}_U, X, X', h(x \oplus y), T_U, \alpha, \omega, r_U, \{ID_U\}\}\)
• \(\text{BEL}(U) = \{\sharp(pw_U), \sharp(b_U), \sharp(r_U), \sharp(\overline{pw}_U), \sharp(X'), \sharp(h(x \oplus y)), \sharp(T_U), \text{LINK}(r_U)\}\)

Then, the global sets are modified as follows:

• Secret set: \(\{pw_U, b_U, \overline{pw}_U, x, y, X', SK_{GS}, h(x \oplus y)\}\)
• Observers sets:
  – Observers(\(X'\)): \(\{U\}\)
  – Observers(\(h(x \oplus y)\)): \(\{U\}\)

After the (U5)–(U16) actions are finished, \(S_n\) starts the actions in BL(\(S_n\)) with the incoming message \(M_1\) from U. The (SN1)–(SN3) actions in BL(\(S_n\)) are performed to verify the correctness of message \(M_1\). If the check succeeds, the (SN4)–(SN8) actions are performed to make the values \(Y, \beta\) and \(r_S\), and to send the message \(M_2\). The local sets of \(S_n\) are changed as follows.

• \(\text{POSS}(S_n) = \{Y, TS, r_S, \beta, SK_{GS}, h(x \oplus y), \{ID_{S_n}\}, \{ID_U, X, T_U, \alpha, \omega\}\text{ from } U\}\)
• \(\text{BEL}(S_n) = \{\sharp(r_S), \sharp(SK_{GS}), \sharp(h(x \oplus y)), \sharp(T_S), \text{LINK}(r_S)\}\)

In this case, the global sets remain unchanged and thus, the secret set is the same as above:

• Secret set: \(\{pw_U, b_U, \overline{pw}_U, x, y, X', SK_{GS}, h(x \oplus y)\}\)

After (SN1)–(SN8) actions of BL(\(S_n\)) are finished, (GW9)–(GW17) actions of BL(\(GW\)) are executed. (GW9)–(GW13) actions check the timestamp of \(S_n\), and then verify the legitimacy of \(U\) and \(S_n\). If they are correct, (GW14)–(GW17) actions of BL(\(S_n\)) are executed to make values(\(\gamma, \delta\)) for authentication and send message. \(\gamma\) is used for authentication with \(S_n\) and \(\delta\) is used for authentication with \(U\).

After the (SN1)–(SN8) actions are done, the (GW9)–(GW13) actions in BL(\(GW\)) are performed to check the legitimacy of \(U\) and \(S_n\). If the verification succeeds, the (GW14)–(GW17) actions are performed to generate \(\gamma\) and \(\delta\) and to send the message \(M_3\) to \(S_n\). The local sets of \(GW\) are modified as shown below.

• \(\text{POSS}(GW) = \{TG, \gamma, \delta, x, y, X', SK_{GS}, h(x \oplus y), \{ID_U, X, T_U, \alpha, \omega, ID_{S_n}, Y, TS, \beta\}\text{ from } S_n\}\)
• \(\text{BEL}(GW) = \{\sharp(x), \sharp(y), \sharp(X'), \sharp(SK_{GS}), \sharp(h(x \oplus y)), \sharp(T_G)\}\)

The global sets are updated as follows:

• Secret set: \(\{pw_U, b_U, \overline{pw}_U, x, y, X', SK_{GS}, h(x \oplus y)\}\)
Observers sets:
- Observers($X'$) = \{GW\}

After the (GW9)–(GW17) actions are finished, the (SN9)-(SN11) actions in BL($S_n$) are conducted to verify the legitimacy of GW and U via the authenticator $\gamma$. If the verification process is completed, the (SN12)–(SN16) actions are performed to generate $\tau$ and $sk$ from $r_S$, $K_{SU}$, $X$ and $Y$, and to send the message $M_4$ to U. The local sets of $S_n$ is updated as follows:

- POSS($S_n$) = \{\{Y, T'_S, K_{SU}, r_S, \tau, sk, SK_{GS}\}, \{T_G, \gamma, \delta\}\} from GW
- BEL($S_n$) = \{\sharp(K_{SU}), \sharp(sk), \sharp(SK_{GS}), \sharp(T'_S), \text{LINK}(r_S)\}

Accordingly, the global sets are modified as follows:

- Secret set: \{pw_U, b_U, pw_U, x, y, K_{SU}, sk, X', SK_{GS}, h(x \oplus y)\}
- Observers sets:
  - Observers($K_{SU}$) = \{S_n\}
  - Observers($sk$) = \{S_n\}

The (U17)–(U19) actions in BL(U) are to check the legitimacy of GW and $S_n$ while the (U20)–(U22) actions are to generate the session key $sk$ from $r_U$, $K_{US}$, $X$ and $Y$. So, the conditions for the linkage rule are satisfied.

- POSS(U) = \{K_{US}, sk, \{ID_U\}, \{Y, T'_S, \delta, \tau\}\} from $S_n$
- BEL(U) = \{\sharp(K_{US}), \sharp(X'), \sharp(\text{LINK}(r_S)), \sharp(h(x \oplus y))\}
- Secret set: \{pw_U, b_U, pw_U, x, y, K_{SU}, K_{US}, sk, X', SK_{GS}, (x \oplus y)\}
- Observers sets:
  - Observers($K_{SU}$) = \{U\}
  - Observers($sk$) = \{U\}

In phase 3, $U$ changes its password and updates $A_U$, $B_U$ and $W_U$ stored in the smart card. In this phase, the local sets of $U$ and the global sets remain unchanged.

The following shows the final version of the global sets.

- Secret set: \{pw_U, b_U, pw_U, x, y, K_{SU}, K_{US}, sk, X', SK_{GS}, h(x \oplus y)\}
- Observers sets:
  - Observers(pw_U) : \{U\}
  - Observers(b_U) : \{U\}
  - Observers(pw_U) : \{U\}
  - Observers(x) : \{GW\}
  - Observers(y) : \{GW\}
  - Observers($K_{SU}$) : \{S_n\}
  - Observers($K_{US}$) : \{U\}
  - Observers($sk$) : \{U, S_n\}
This result implies that:

- $pw_U$, $b_U$ and $\overline{pw}_U$ are known only to the user $U$.
- $x$ and $y$ are known only to the gateway $GW$.
- The long-term key $SK_{GS}$ shared between $S_n$ and $GW$ is not exposed.
- $X'$ is only known to $U$ and $GW$.
- $K_{US}$ and $K_{SU}$ are only available to $U$ and $S_n$.
- The session key $sk$ is securely shared between $U$ and $S_n$.
- $h(x \oplus y)$ is only known to the authorized principals: $U$, $S_n$ and $GW$.
- $U$, $S_n$ and $GW$ are mutually authenticated during the protocol execution.

This verifies the security claims we made in the previous subsection.

8. Conclusions

In this paper, we have identified that Shi et al.’s ECC-based authentication protocol designed for wireless sensor networks (WSNs) is vulnerable to: a session key attack, a stolen smart card attack, and a sensor energy exhausting attack. We have also proposed a new authentication protocol that addresses the identified security weaknesses. Our proposed protocol is as efficient as Shi et al.’s protocol and is better suited for WSNs than Yeh et al.’s protocol, the predecessor of Shi et al.’s protocol. As for the security of the proposed protocol, we have provided a heuristic analysis and formally verified the analysis using Rubin logic.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT&Future Planning (2014R1A1A2002775).

Author Contributions

Younsung Choi, Donghoon Lee, Jiye Kim, Jaewook Jung, Junghyun Nam and Dongho Won have contributed to security analysis, design of the proposed scheme, and manuscript preparation.

Conflicts of Interest

The authors declare no conflict of interest.
References

1. Dressler, F. Authenticated reliable and semi-reliable communication in wireless sensor networks. *Int. J. Netw. Secur.* 2008, 7, 61–68.
2. Yeh, H.L.; Chen, T.H.; Liu, P.C.; Kim, T.H.; Wei, H.W. A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors* 2011, 11, 4767–4779.
3. Shi, W.; Gong, P. A new user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Int. J. Distrib. Sens. Netw.* 2013, 2013, 730831.
4. Satpute, R.S.; Thakare, A.N. Survey on Security in Wireless Sensor Networks Using Elliptical Curves Cryptography. Available online: http://www.ijert.org/view.php?id=5878&title=survey-on-security-in-wireless-sensor-networks-using-elliptical-curves-cryptography (accessed on 6 June 2014).
5. Watro, R.; Kong, D.; Cuti, S.F.; Gardiner, C.; Lynn, C.; Krus, P. TinyPK: Securing Sensor Networks with Public Key Technology. In *ACM Workshop on Security of Ad Hoc and Sensor Networks*; ACM Press: Washington, DC, USA, 2004; pp. 59–64.
6. Wong, K.H.; Zheng, Y.; Cao, J.; Wang, S. A dynamic user authentication scheme for wireless sensor networks. In *Proceedings of 2006 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, Taichung, Taiwan, 5–7 June 2006; pp. 1–9.
7. Tseng, H.R.; Jan, R.H.; Yang, W. An improved dynamic user authentication scheme for wireless sensor networks. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM ’07)*, Washington, DC, USA, 26–30 November 2007; pp. 986–990.
8. Das, M.L. Two-factor user authentication in wireless sensor networks. *IEEE Trans. Wirel. Commun.* 2009, 8, 1086–1090.
9. He, D.; Gao, Y.; Chan, S.; Chen, C.; Bu, J. An enhanced two-factor user authentication scheme in wireless sensor networks. *Ad Hoc Sens. Wirel. Netw.* 2010, 10, 361–371.
10. Khan, M.K.; Alghathbar, K. Cryptanalysis and security improvements of “two-factor user authentication in wireless sensor networks”. *Sensors* 2010, 10, 2450–2459.
11. Chen, T.H.; Shih, W.K. A robust mutual authentication protocol for wireless sensor networks. *ETRI J.* 2010, 32, 704–712.
12. Han, W. Weakness of a Secured Authentication Protocol for Wireless Sensor Networks Using Elliptic Curves Cryptography. Available online: http://eprint.iacr.org/2011/293 (accessed on 27 June 2011).
13. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, Santa Barbara, CA, USA, 15–19 August 1999; pp. 388–397.
14. Messerges, T.S.; Dabbish, E.A.; Sloan, R.H. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* 2002, 51, 541–552
15. Muthukuru, J.; Sathyanarayana, B. A Survey of Elliptic Curve Cryptography Implementation Approaches for Efficient Smart Card Processing. *Glob. J. Comput. Sci. Technol.* 2012, 12, 7–12.
16. Kar, J.; Majhi, B. An efficient password security of multiparty key exchange protocol based on ECDLP. In *Proceedings of 2010 2nd International Conference on Computer Engineering and Technology (ICCET)*, Chengdu, China, 16–18 April 2010; pp. 405–413.
17. Venkata, C.; Saikat, C.; Mukesh, S. A distributed multi-party key agreement protocol for dynamic collaborative groups using ECC. *J. Parall. Distrib. Comput.* 2006, 66, 959–970.

18. Lu, R.; Cao, Z.; Chai, Z.; Liang, X. A simple user authentication scheme for grid computing. *Int. J. Netw. Secur.* 2008, 7, 202–206.

19. Nam, J.; Paik, J.; Kang, H.K.; Kim, U.M.; Won, D. An off-line dictionary attack on a simple three-party key exchange protocol. *IEEE Commun. Lett.* 2009, 13, 205–207.

20. Lee, Y.; Kim, S.; Won, D. Enhancement of two-factor authenticated key exchange protocols in public wireless LANs. *Comput. Electr. Eng.* 2010, 36, 213–223.

21. Nam, J.; Choo, K.K.R.; Kim, M.; Paik, J.; Won, D. Dictionary attacks against password-based authenticated three-party key exchange protocols. *KSII Trans. Internet Inf. Syst.* 2013, 7, 3244–3260.

22. Jeon, W.; Kim, J.; Nam, J.; Lee, Y.; Won, W. An enhanced secure authentication scheme with anonymity for wireless environments. *IEICE Trans. Commun.* 2012, E95.B, 2505–2508.

23. Buttyan, L.; Csik, L. Security analysis of reliable transport layer protocols for wireless sensor networks. In Proceedings of 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), Mannheim, Germany, 29 March–2 April 2010; pp. 419–424.

24. Brownfield, M.; Gupta, Y.; Davis, N. Wireless sensor network denial of sleep attack. In Proceedings of 2005 the Sixth Annual IEEE SMC Information Assurance Workshop (IAW ’05), West Point, NY, USA, 15–17 June 2005; pp. 356–364.

25. Khouzani, M.H.R.; Sarkar, S. Maximum damage battery depletion attack in mobile sensor networks. *IEEE Trans. Autom. Control* 2011, 56, 2358–2368.

26. Khouzani, M.H.R.; Sarkar, S.; Altman, E. Maximum damage malware attack in mobile wireless networks. *ACM Trans. Netw.* 2012, 20, 1347–1360.

27. Chen, L.; Cheng, Z.; Smart, N.P. Identity-based key agreement protocols from pairings. *Int. J. Inf. Secur.* 2007, 6, 213–241.

28. Cao, X.; Zeng, X.; Kou, W.; Hu, L. Identity-based anonymous remote authentication for value-added services in mobile networks. *IEEE Trans. Veh. Technol.* 2009, 58, 3508–3517.

29. He, D.; Chen, J.; Hu, J. An ID-based client authentication with key agreement protocol for mobile client-server environment on ECC with provable security. *Inf. Fusion* 2012, 13, 223–230.

30. Jeong, H.; Won, D.; Kim, S. Weaknesses and improvement of secure hash-based strong-password authentication protocol. *J. Inf. Sci. Eng.* 2010, 26, 1845–1858.

31. Lee, C.C.; Hwang, M.S.; Liao, I.E. Security enhancement on a new authentication scheme with anonymity for wireless environments. *IEEE Trans. Ind. Electron.* 2006, 53, 1683–1687.

32. Nam, J.; Kim, S.; Park, S.; Won, D. Security analysis of a nonce-based user authentication scheme using smart cards. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 2007, E90-A, 299–302.

33. Rubin, A.D.; Honeyman, P. Nonmonotonic cryptographic protocols. In Proceedings of the Computer Security Foundations Workshop VII, 1994 (CSFW 7), Franconia, NH, USA, 14–16 June 1994; pp. 100–116.
34. Das, M.L.; Narasimhan, V.L. Towards a formal verification of an authentication protocol using non-monotonic logic. In Proceedings of the IEEE Fifth International Conference on Information Technology: New Generations, Las Vegas, NV, USA, 7–9 April 2008; pp. 545–550.

35. Xu, Y.; Xie, X. Analysis of authentication protocols based on Rubin logic. In Proceedings of the 4th IEEE International Conference on Wireless Communications, Networking and Mobile Computing, Dalian, China, 12–14 October 2008; pp. 1–5.

36. Vaidya, B.; Makrakis, D.; Mouftah, H. Two-factor mutual authentication with key agreement in wireless sensor networks. *Secur. Commun. Netw.* **2012**, doi:10.1002/sec.517.

© 2014 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).