






Article

WSN-SLAP: Secure and Lightweight Mutual Authentication Protocol for Wireless Sensor Networks

Deok Kyu Kwon ¹, Sung Jin Yu ¹, Joon Young Lee ¹, Seung Hwan Son ¹ and Young Ho Park ^{1,2,*}

¹ School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Korea; kdk145@knu.ac.kr (D.K.K.); darkskiln@knu.ac.kr (S.J.Y.); harry250@knu.ac.kr (J.Y.L.); sonshawn@knu.ac.kr (S.H.S.)

² School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea

* Correspondence: parkyh@knu.ac.kr; Tel.: +82-53-950-7842

Abstract: Wireless sensor networks (WSN) are widely used to provide users with convenient services such as health-care, and smart home. To provide convenient services, sensor nodes in WSN environments collect and send the sensing data to the gateway. However, it can suffer from serious security issues because susceptible messages are exchanged through an insecure channel. Therefore, secure authentication protocols are necessary to prevent security flaws in WSN. In 2020, Moghadam et al. suggested an efficient authentication and key agreement scheme in WSN. Unfortunately, we discover that Moghadam et al.'s scheme cannot prevent insider and session-specific random number leakage attacks. We also prove that Moghadam et al.'s scheme does not ensure perfect forward secrecy. To prevent security vulnerabilities of Moghadam et al.'s scheme, we propose a secure and lightweight mutual authentication protocol for WSNs (WSN-SLAP). WSN-SLAP has the resistance from various security drawbacks, and provides perfect forward secrecy and mutual authentication. We prove the security of WSN-SLAP by using Burrows-Abadi-Needham (BAN) logic, Real-or-Random (ROR) model, and Automated Verification of Internet Security Protocols and Applications (AVISPA) simulation. In addition, we evaluate the performance of WSN-SLAP compared with existing related protocols. We demonstrate that WSN-SLAP is more secure and suitable than previous protocols for WSN environments.

Keywords: mutual authentication; wireless sensor networks; BAN logic; ROR model; AVISPA



Citation: Kwon, D.K.; Yu, S.J.; Lee, J.Y.; Son, S.H.; Park, Y.H. WSN-SLAP: Secure and Lightweight Mutual Authentication Protocol for Wireless Sensor Networks. *Sensors* **2021**, *21*, 936. <https://doi.org/10.3390/s21030936>

Academic Editor: Jaime Lloret Mauri

Received: 8 January 2021

Accepted: 27 January 2021

Published: 30 January 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 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 (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As a rapid development of wireless communication technology, wireless sensor networks (WSN) can be applied to various environments such as smart grids, smart homes, agriculture, industrial internet of things (IoT), and health-care [1–5]. People can achieve a more bountiful life by utilizing WSN environments. Generally, WSN environments consist of sensor nodes, a gateway, and users, as shown in Figure 1. Sensor nodes detect and monitor their surrounding environment. Then, sensor nodes transmit the monitored data to the gateway. The gateway relays and analyzes the message between sensor nodes and users. The gateway also manages the private information of sensor nodes and users to provide secure services. Users can access the data collected by sensor nodes through the gateway.

An example of the application environment in WSN is health-care services. Wearable sensors attached to a patient analyze the health condition of the patient. Then, these sensors send the collected data to the physician. However, these services can be exposed to various security attacks because each entity exchanges information through a public channel. If an adversary intercepts messages in WSN, the adversary can disguise as a legal user and send an incorrect message to the sensor node. Moreover, if an adversary registers to the gateway as a legal entity, the adversary can try to obtain other legal user's sensitive information.

Therefore, we need an authentication protocol that can provide secure services and prevent various attacks in WSN environments.

In 2020, Moghadam et al. [6] suggested an authentication and key agreement scheme for WSN environments utilizing Elliptic-Curve Diffie-Hellman (ECDH) [7]. They demonstrated that their scheme is efficient and secure against various security attacks such as replay, password guessing, stolen verifier, and man-in-the-middle (MITM) attacks. However, we discover that Moghadam et al.'s scheme does not provide security against insiders, and session-specific random number leakage attacks. We also prove that Moghadam et al.'s scheme does not support perfect forward secrecy. Moreover, each entity performs Elliptic Curve Cryptography (ECC) multiplication operations to compute a session key in Moghadam et al.'s scheme. However, ECC requires heavy computational costs. Since sensor nodes have low computation capabilities and storage resources in a WSN environment, we cannot ensure real-time communications using ECC in WSN environments. Therefore, using Moghadam et al.'s scheme makes it difficult to provide efficient services. To improve security vulnerabilities and reduce the computational cost of Moghadam et al.'s scheme, we propose a secure and lightweight mutual authentication protocol (WSN-SLAP) considering security and efficiency features using hash functions and XOR operations.

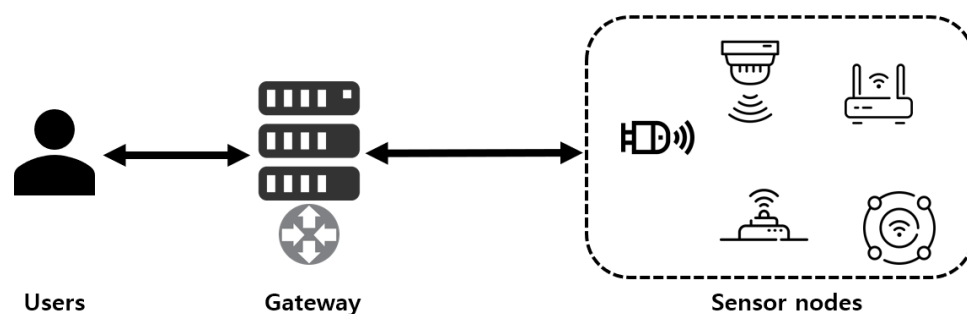


Figure 1. System model in Wireless sensor networks (WSNs).

1.1. Contributions

Our paper's contributions are as below.

- We analyze and prove the security vulnerabilities of Moghadam et al.'s scheme. Then, we propose WSN-SLAP to resolve security vulnerabilities of Moghadam et al.'s scheme.
- We demonstrate the mutual authentication of WSN-SLAP using Burrows–Abadi–Needham (BAN) logic [8].
- We prove the session key security of WSN-SLAP by using the Real-or-Random (ROR) model [9].
- We use Automated Verification of Internet Security Protocols and Applications (AVISPA) [10,11] to prove security features of WSN-SLAP against replay and MITM attacks.
- We analyze the communication cost, the computational cost, and security properties of WSN-SLAP compared with related schemes.

1.2. Adversary Model

WSN-SLAP uses a well-known adversary model called the Dolev–Yao (DY) model [12]. Through the DY model, the adversary can eavesdrop, delete, intercept, and insert exchanged messages through a public channel. Moreover, the adversary can get exposed session-specific ephemeral parameters, which is based on the Canetti–Krawczyk (CK) adversary model [13]. The adversary can perform various security attacks with the DY model and the CK model. The detailed assumptions of the adversary model are defined in the following manner.

- If an adversary registers as a legal user to the gateway, the adversary can authenticate with other entities.

- An adversary can obtain a user's lost/stolen smart card. The adversary can perform the power analysis attack [14] to get stored parameters of the smart card.
- An adversary can attempt various attacks such as replay, sensor node capture, stolen verifier, and off-line password guessing attacks.

1.3. Organization

In Section 2, we describe related works for WSN environments. Then, we revisit Moghadam et al.'s scheme in Section 3 and prove the security flaws of Moghadam et al.'s scheme in Section 4. Section 5 illustrates WSN-SLAP. In Section 6, we perform informal and formal security analyses of WSN-SLAP by using BAN logic, the ROR model, and AVISPA simulation tool. In Section 7, we analyze WSN-SLAP's performance compared with the existing related protocols. In Section 8, we conclude and summarize our paper.

2. Related Works

In the past few decades, numerous password-based authentication schemes have been proposed to provide security and efficiency in WSN environments [15–19]. In 1981, Lamport [20] suggested an authentication mechanism based on a password. Lamport used one-way hash functions to encode the password and stored the hashed password inside the system. In 2006, Wong et al. [21] suggested a password-based authentication scheme in WSN environments. Unfortunately, Tseng et al. [22] proved that Wong et al.'s scheme is insecure against forgery and replay attacks. Tseng et al. demonstrated a dynamic user authentication scheme to improve security vulnerabilities of Wong et al. [21]'s scheme. However, these schemes [20–22] can suffer from on/off-line password guessing attacks because they only used the password as a factor to login and authenticate with other entities.

In the last few decades, two-factor-based authentication schemes [23–25] have been presented using hash functions and XOR operations to improve single factor's security weaknesses. In 2009, Das et al. [23] proposed a two-factor authentication scheme based on a smart card in WSNs. They demonstrated that their scheme can prevent various attacks such as replay, stolen verifier, and off-line password guessing attacks. However, Khan et al. [24] analyzed that Das et al. [23]'s scheme is vulnerable to privileged insider attack. He et al. [25] found that Das et al. [23]'s scheme is vulnerable to insider and impersonation attacks. To improve the security vulnerabilities of Das et al.'s scheme, He et al. [25] suggested an enhanced two-factor user authentication scheme for WSNs. However, these schemes [23–25] can suffer from various attacks such as those using stolen smart cards and mobile devices.

To resolve the security flaws associated with two-factor-based authentication schemes and improve the security level in WSN environments, researchers have proposed many ECC-based authentication schemes [26–31]. In 2011, Yeh et al. [26] proposed an authentication protocol for WSN environments using ECC. Yeh et al.'s scheme used a smart card and ECC to prevent various security issues such as insider, and masquerade attacks. Choi et al. [27] suggested an ECC-based user authentication scheme for WSN. However, Wu et al. [28] pointed out that Choi et al.'s protocol does not provide security against forgery attack. Nam et al. [29] suggested a secure authentication protocol for WSN based on ECC. Nam et al.'s scheme provides a secure protocol based on an Elliptic Curve Computation Diffie-Hellman (ECCDH) problem. In 2016, Jiang et al. [30] proposed an ECC-based authentication scheme. Jiang et al.'s scheme provides secure communications and untraceability in WSN environments. In 2017, Wu et al. [31] suggested a user authentication scheme using ECC. Wu et al.'s scheme can preserve user privacy in WSN environments. However, sensor nodes in WSN have low computing power and resources. Therefore, it is difficult to provide efficiency in WSN environments using these schemes [26–31] because ECC requires large computational resources.

In 2020, Moghadam et al. [6] suggested an authentication and key agreement scheme using ECDH. They asserted that their scheme provides resistance against various attacks

such as replay, MITM, off-line password guessing, and stolen verifier attacks. However, we discover that Moghadam et al.’s scheme is vulnerable to insider, session-specific random number leakage attacks and perfect forward secrecy. Moreover, Moghadam et al.’s scheme suffers from heavy computational cost because it involves an ECC-based computation. Therefore, we propose WSN-SLAP, which has resistance to various security problems.

3. Review of Moghadam et al.’s Scheme

Moghadam et al. proposed an authentication scheme based on ECDH in WSN [6]. Moghadam et al.’s scheme is composed of sensor node registration, user registration, and login and authentication phases. Table 1 indicates the notations of Moghadam et al.’s scheme and WSN-SLAP.

Table 1. Notations.

Notation	Description
U_i	User
GW	Gateway
S_j	Sensor node
ID_i	Real identity of user
PW_i	Password of user
PID_i	Pseudo identity of user
SID_j	Real identity of sensor node
k_{GWN}	Master key of gateway
KG	Shared secret key between gateway and sensor node
X	Public key of gateway
G	Elliptic curve group
P	Generator of G
$R_k, N_k, z_i, a_i, f_i, g_i, q_i$	Random numbers
T_k	Timestamp
SK	Session key
E_k/D_k	Symmetric key encryption/decryption
$h(.)$	Hash function
$ $	Concatenation function
\oplus	Exclusive-or function

3.1. Sensor Node Registration Phase

In this phase, a sensor node S_j sends its identity to the gateway GW . Then, GW computes a shared secret parameter between GW and S_j . In Figure 2, we show the sensor node registration phase and the details are as follows.

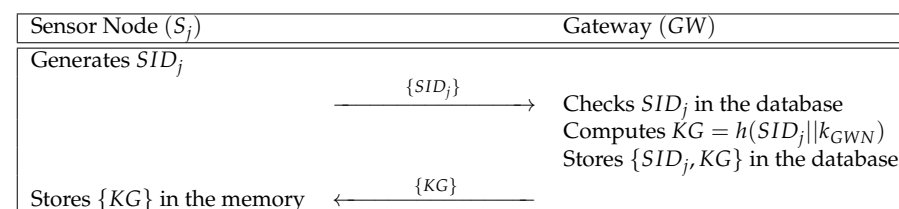


Figure 2. Sensor node registration phase of Moghadam et al.’s scheme.

- Step 1:** S_j generates its identity SID_j , and sends it to GW over a secure channel.
- Step 2:** GW receives SID_j and checks the validity of SID_j . After that, GW computes $KG = h(SID_j || k_{GWN})$, and stores $\{SID_j, KG\}$ in its secure database, where k_{GWN} is the master key of GW . Finally, GW sends $\{KG\}$ to S_j .
- Step 3:** S_j receives and stores $\{KG\}$ in its database.

3.2. User Registration Phase

A user U_i registers to the gateway GW by sending an identity and a masked password value. Then, GW issues a smart card to U_i . In Figure 3, we describe the user registration phase and the details are shown as below.

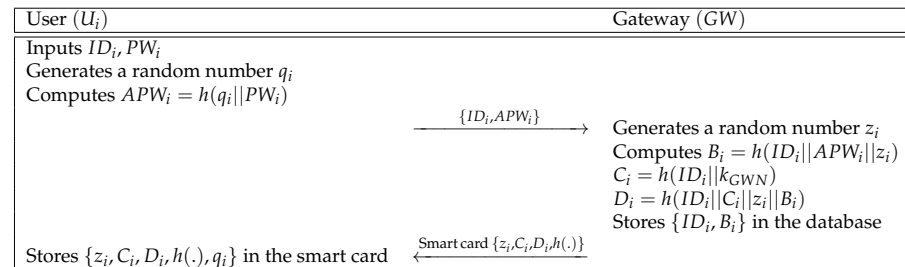


Figure 3. User registration phase of Moghadam et al.'s scheme.

- Step 1:** U_i inputs the identity ID_i and the password PW_i , and then generates a random number q_i . After that, U_i computes $APW_i = h(q_i || PW_i)$ and sends the registration request message $\{ID_i, APW_i\}$ to the gateway GW over a secure channel.
- Step 2:** GW receives $\{ID_i, APW_i\}$ from U_i , and then generates a random number z_i . After that, GW computes $B_i = h(ID_i || APW_i || z_i)$, $C_i = h(ID_i || k_{GWN})$, and $D_i = h(ID_i || C_i || z_i || B_i)$. Finally, GW stores $\{z_i, C_i, D_i, h(\cdot)\}$ in a smart card and issues it to U_i over a secure channel.
- Step 3:** U_i receives the smart card, and stores q_i in the smart card. Finally, parameters $\{z_i, C_i, D_i, h(\cdot), q_i\}$ are stored in the smart card.

3.3. Login and Authentication Phase

After the registration phase, the user U_i authenticates the gateway GW . In Figure 4, we describe the login and authentication phase and the detailed steps of the phase are shown as below.

- Step 1:** After inserting the smart card, U_i inputs the identity ID_i^* and the password PW_i^* . The smart card computes $APW_i^* = h(PW_i^* || q_i)$, $B_i^* = h(ID_i^* || APW_i^* || z_i)$, $D_i^* = h(ID_i^* || C_i || z_i || B_i^*)$ and verifies $D_i^* \stackrel{?}{=} D_i$. If the verification process is successful, the smart card generates a random nonce a_i and timestamp T_1 . With the public key of the gateway X , the smart card computes $A_1 = a_i \cdot P$, $A_2 = a_i \cdot X$, $DID_i = ID_i \oplus A_{2(x)}$, $A_3 = SID_j \oplus A_{2(x)}$, and $A_4 = E_{A_2}(B_i || SID_j || A_3)$. At last, the smart card sends $\{A_1, A_3, A_4, T_1\}$ to GW through a public channel.
- Step 2:** GW receives $\{A_1, A_3, A_4, T_1\}$ from U_i , and selects a timestamp T_2 and checks the validity of T_1 . If the timestamp is valid, GW computes $A_2 = k_{GWN} \cdot A_1$, $D_{A_2}(A_4) = (B_i^* || SID_i^* || A_3^*)$, $A_3 = SID_i^* \oplus A_{2(x)}$ and verifies $A_3^* \stackrel{?}{=} A_3$. If the equality holds, GW generates a random nonce g_i and computes $KG = h(SID_j || k_{GWN})$, $D_1 = KG \oplus A_2$, $D_2 = h(A_2 || SID_j || A_3)$. At last, GW sends $\{g_i \cdot P, D_1, D_2, T_2\}$ to the sensor node S_j over a public channel.
- Step 3:** After reception of the message $\{g_i \cdot P, D_1, D_2, T_2\}$ from GW , S_j selects a timestamp T_3 and checks the validity of T_2 . Then, S_j computes $A_2 = KG \oplus D_1$, $A_3 = SID_j \oplus A_{2(x)}$, $D_2^* = h(A_2 || SID_j || A_3)$ and verifies $D_2^* \stackrel{?}{=} D_2$. If the verification is legitimate, S_j generates a random nonce f_i , and computes $sk = h(A_2 || f_i \cdot g_i \cdot P)$, $X_i = h(sk || KG)$. At last, S_j sends $\{f_i \cdot P, X_i, T_3\}$ to GW .
- Step 4:** After receiving $\{f_i \cdot P, X_i, T_3\}$ from S_j , GW selects a timestamp T_4 and checks the validity of T_3 . Then, GW computes $sk = h(A_2 || f_i \cdot g_i \cdot P)$, $X_i = h(sk || KG)$ and verifies $X_i^* \stackrel{?}{=} X_i$. If it is equal, GW computes $D_4 = E_{A_2}(g_i)$, $y_i = h(sk || A_3)$ and sends $\{y_i, D_4, T_4\}$ to U_i .

Step 5: U_i receives the message $\{y_i, D_4, T_4\}$, and selects a timestamp T_5 and checks the validity of T_4 . At last, U_i computes $D_{A_2}(D_4) = (g_i), sk = h(A_2||f_i \cdot g_i \cdot P), y_i^* = h(sk||A_3)$ and verifies $y_i^* \stackrel{?}{=} y_i$. If it is equal, the key agreement is successful.

User (U_i)	Gateway (GW)	Sensor Node (S_j)
Inserts the smart card Inputs ID_i^*, PW_i^* Computes $APW_i^* = h(PW_i^* q_i)$ $B_i^* = h(ID_i^* APW_i^* z_i)$ $D_i^* = h(ID_i^* C_i z_i B_i^*)$ Checks $D_i^* \stackrel{?}{=} D_i$ Generates a random nonce a_i Computes $A_1 = a_i \cdot P, A_2 = a_i \cdot X$ $DD_i = ID_i \oplus A_{2(x)}$ $A_3 = SID_j \oplus A_{2(x)}$ $A_4 = E_{A_2}(B_i SID_j A_3)$ $\{A_1, A_3, A_4, T_1\}$	Selects a timestamp T_2 Checks $ T_2 - T_1 \leq \Delta T$ Computes $A_2 = k_{GW} \cdot A_1$ $D_{A_2}(A_4) = (B_i^* SID_j^* A_3^*)$ $A_3 = SID_j^* \oplus A_{2(x)}$ Checks $A_3^* \stackrel{?}{=} A_3$ Generates a random nonce g_i Computes $KG = h(SID_j k_{GW})$ $D_1 = KG \oplus A_2$ $D_2 = h(A_2 SID_j A_3)$ $\{g_i \cdot P, D_1, D_2, T_2\}$	Selects a timestamp T_3 Checks $ T_3 - T_2 \leq \Delta T$ Computes $A_2 = KG \oplus D_1$ $A_3 = SID_j \oplus A_{2(x)}$ $D_2^* = h(A_2 SID_j A_3)$ Checks $D_2^* \stackrel{?}{=} D_2$ Generates a random nonce f_i Computes $sk = h(A_2 f_i \cdot g_i \cdot P)$ $X_i = h(sk KG)$ $\{f_i \cdot P, X_i, T_3\}$
Selects a timestamp T_5 Checks $ T_5 - T_4 \leq \Delta T$ Computes $D_{A_2}(D_4) = (g_i)$ $sk = h(A_2 f_i \cdot g_i \cdot P)$ $y_i^* = h(sk A_3)$ Checks $y_i^* \stackrel{?}{=} y_i$	Selects a timestamp T_4 Checks $ T_4 - T_3 \leq \Delta T$ Computes $sk = h(A_2 f_i \cdot g_i \cdot P)$ $X_i = h(sk KG)$ Checks $X_i^* \stackrel{?}{=} X_i$ Computes $D_4 = E_{A_2}(g_i)$ $y_i = h(sk A_3)$ $\{y_i, D_4, T_4\}$	

Figure 4. Login and authentication phase of Moghadam et al.'s scheme.

4. Cryptanalysis of Moghadam et al.'s Scheme

In this section, we demonstrate the security vulnerabilities of Moghadam et al.'s scheme [6] such as insider, and session-specific random number leakage attacks. Moghadam et al.'s scheme also does not achieve perfect forward secrecy.

4.1. Insider Attack

If an adversary \mathcal{A} ordinary registers as a legal user U_i , \mathcal{A} can authenticate with the gateway GW and the sensor node S_j by exchanging messages. With this information, \mathcal{A} can compute another legal user U_i^l 's session key. The details are shown as below.

Step 1: \mathcal{A} inserts the smart card, and inputs the identity ID_i and the password PW_i of \mathcal{A} . Then, the smart card checks the validity of \mathcal{A} , and sends a login request message $\{A_1, A_3, A_4, T_1\}$ to GW . After authenticating \mathcal{A} , GW sends $\{g_i \cdot P, D_1, D_2, T_2\}$ to S_j . Upon reception of the message $\{g_i \cdot P, D_1, D_2, T_2\}$, S_j computes a session key sk . Then, S_j sends the authentication response message $\{f_i \cdot P, X_i, T_3\}$ to GW . GW computes

the session key and sends $\{y_i, D_4, T_4\}$ to \mathcal{A} . \mathcal{A} computes the session key and obtains communication messages during the login and authentication phase.

Step 2: After obtaining the message $\{g_i \cdot P, D_1, D_2, T_2\}$, \mathcal{A} computes $KG = D_1 \oplus A_2$, where A_2 is the secret key of \mathcal{A} using ECC and KG is a shared secret key between GW and S_j .

Step 3: \mathcal{A} intercepts a message $\{g_i^l \cdot P, D_1^l, D_2^l, T_2^l\}$ from the message of another legal user U_i^l . Since \mathcal{A} knows KG , it can compute $A_2^l = D_1^l \oplus KG$, where A_2^l is the secret key of U_i^l .

Step 4: \mathcal{A} obtains the message $\{y_i^l, D_4^l, T_4^l\}$ and decrypts D_4^l using the secret key A_2^l of U_i^l . Then, \mathcal{A} can obtain the random secret nonce g_i^l of sensor node. \mathcal{A} can compute $f_i^l \cdot g_i^l \cdot P$ by utilizing the message $\{f_i^l \cdot P, X_i^l, T_3^l\}$. Finally, \mathcal{A} compute the session key $sk^l = h(A_2^l || f_i^l \cdot g_i^l \cdot P)$.

Therefore, Moghadam et al.'s scheme cannot prevent insider attacks.

4.2. Perfect Forward Secrecy

Moghadam et al. demonstrated that their scheme can ensure the security feature of perfect forward secrecy. However, if the adversary \mathcal{A} gets the master key k_{GWN} of the gateway GW , the adversary can compute the legal user U_i 's session key sk . The details are shown in following steps.

Step 1: If \mathcal{A} obtains the master key k_{GWN} , \mathcal{A} can compute the secret key $A_2 = k_{GWN} \cdot A_1$ of U_i by utilizing the login request message $\{A_1, A_3, A_4, T_1\}$.

Step 2: When \mathcal{A} intercepts the message $\{y_i, D_4, T_4\}$, \mathcal{A} can decrypt $E_{A_2}(g_i)$ because A_2 is the symmetric key between the U_i and the gateway GW .

Step 3: After \mathcal{A} obtains the message $\{f_i \cdot P, X_i, T_3\}$, \mathcal{A} can get (A_2, g_i) and $(f_i \cdot P)$. At last, \mathcal{A} computes U_i 's session key $sk = h(A_2 || f_i \cdot g_i \cdot P)$.

Consequently, Moghadam et al.'s scheme does not ensure perfect forward secrecy.

4.3. Session-Specific Random Number Leakage Attack

Suppose that a random nonce a_i is disclosed to an adversary \mathcal{A} . Using the public key X of the gateway GW , \mathcal{A} can calculate $A_2 = a_i \cdot X$. Then, \mathcal{A} can compute the session key sk . The details are described as below.

Step 1: After getting the parameter A_2 , \mathcal{A} captures the message $\{y_i, D_4, T_4\}$. Then, \mathcal{A} decrypts $D_4 = E_{A_2}(g_i)$ by using the symmetric key A_2 and obtains g_i .

Step 2: \mathcal{A} eavesdrops the message of the sensor node S_j $\{f_i \cdot P, X_i, T_3\}$. Finally, \mathcal{A} computes the session key $sk = h(A_2 || f_i \cdot g_i \cdot P)$ using $f_i \cdot P$ in the message of S_j .

Therefore, Moghadam et al.'s scheme cannot prevent session-specific random number leakage attacks.

5. Proposed Scheme

We propose a secure and lightweight mutual authentication protocol for WSN environments to resolve security weaknesses of Moghadam et al.'s scheme [6]. To consider the resource-limited sensor nodes, WSN-SLAP uses hash functions and XOR operations that generate low computational overheads. WSN-SLAP is composed of sensor node registration, user registration, login and authentication, password update, and sensor node addition phases.

5.1. Sensor Node Registration Phase

If a sensor node S_j sends a registration request message, the gateway GW computes a secret parameter for the sensor node. Then, S_j stores the parameter. We show the sensor node registration phase in Figure 5 and the details are presented as below.

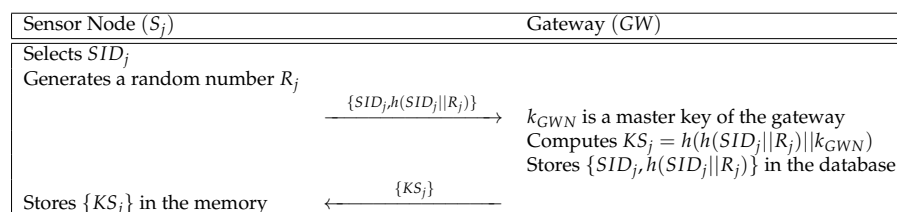


Figure 5. Sensor node registration phase of a secure and lightweight mutual authentication protocol (WSN-SLAP).

Step 1: S_j selects its identity SID_j and generates a random number R_j . Then, S_j computes $h(SID_j||R_j)$ and sends $\{SID_j, h(SID_j||R_j)\}$ to GW over a secure channel.

Step 2: GW receives $\{SID_j, h(SID_j||R_j)\}$ and computes $KS_j = h(h(SID_j||R_j)||k_{GWN})$, where k_{GWN} is the master key of GW . GW stores $\{SID_j, h(SID_j||R_j)\}$ in the secure database and sends $\{KS_j\}$ to S_j .

Step 3: At last, S_j stores $\{KS_j\}$ in its memory.

5.2. User Registration Phase

A user U_i sends a registration request message to the gateway GW . Then, GW computes secret parameters and issues a smart card to the user. In Figure 6, we describe the user registration phase and the detailed steps are shown as below.

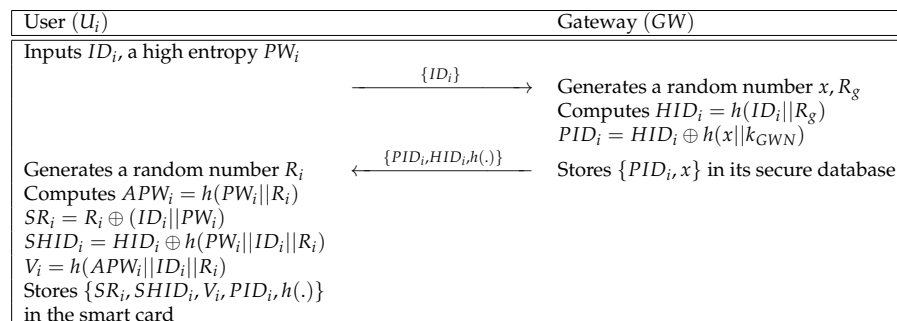


Figure 6. User registration phase of WSN-SLAP.

Step 1: U_i inputs an identity ID_i and a high entropy password PW_i . After that, U_i transmits $\{ID_i\}$ to GW via a secure channel.

Step 2: GW generates random numbers x and R_g , and computes $HID_i = h(ID_i||R_g)$, $PID_i = HID_i \oplus h(x||k_{GWN})$. GW stores $\{PID_i, x\}$ in its secure database and sends the message $\{PID_i, HID_i, h(\cdot)\}$ to U_i .

Step 3: U_i generates a random number R_i . With R_i , U_i computes $APW_i = h(PW_i||R_i)$, $SR_i = R_i \oplus (ID_i||PW_i)$, $SHID_i = HID_i \oplus h(PW_i||ID_i||R_i)$, and $V_i = h(APW_i||ID_i||R_i)$. Finally, U_i stores $\{SR_i, SHID_i, V_i, PID_i, h(\cdot)\}$ in the smart card.

5.3. Login and Authentication Phase

To access information of the sensor S_j , the user U_i sends a login request message to the gateway GW . In Figure 7, we describe the login and authentication phase and the details are presented below.

User (U_i)	Gateway (GW)	Sensor Node (S_j)
Inserts the smart card Inputs ID_i, PW_i Computes $R_1^* = SR_i \oplus h(ID_i PW_i)$ $APW_i^* = h(PW_i R_i)$ $V_i^* = h(APW_i ID_i R_1^*)$ Checks $V_i^* \stackrel{?}{=} V_i$ Generates a random nonce N_1 Computes $HID_i = SHID_i \oplus h(PW_i ID_i R_i)$ $S_i = SID_j \oplus h(PID_i HID_i)$ $M_1 = N_1 \oplus h(HID_i PID_i)$ $V_1 = h(SID_j PID_i N_1 HID_i)$ $\{PID_i, S_i, M_1, V_1\}$	Retrieves PID_i and the secret value x Computes $HID_i^* = PID_i \oplus h(x k_{GWN})$ $SID_j^* = S_i \oplus h(PID_i HID_i^*)$ $N_1^* = M_1 \oplus h(HID_i^* PID_i)$ $V_1^* = h(SID_j^* PID_i N_1^* HID_i^*)$ Checks $V_1^* \stackrel{?}{=} V_1$ Generates a random nonce N_2 Retrieves SID_j and $h(SID_j R_j)$ Computes $KS_j = h(h(SID_j R_j) k_{GWN})$ $M_2 = h(N_2 HID_i) \oplus h(KS_j PID_i)$ $M_3 = N_1 \oplus h(h(N_2 HID_i) KS_j)$ $V_2 = h(PID_i SID_j h(N_2 HID_i) N_1)$ $\{PID_i, M_2, M_3, V_2\}$	Computes $h(N_2 HID_i)^* = M_2 \oplus h(KS_j PID_i)$ $N_1^* = M_3 \oplus h(h(N_2 HID_i)^* PID_i)$ $V_2^* = h(PID_i SID_j h(N_2 HID_i) N_1^*)$ Checks $V_2^* \stackrel{?}{=} V_2$ Generates a random nonce N_3 Computes $SK = h(h(N_2 HID_i) N_3 N_1)$ $M_4 = N_3 \oplus h(KS_j N_2)$ $V_3 = h(SK N_3 SID_j)$ $\{M_4, V_3\}$
Computes $PID_i^{new} = P_i \oplus h(N_1 HID_i)$ $N_2^* = M_5 \oplus h(HID_i SID_j N_1)$ $N_3^* = M_6 \oplus h(N_2^* HID_i PID_i^{new})$ $SK^* = h(h(N_2^* HID_i) N_3^* N_1)$ $V_4^* = h(N_2^* N_3^* PID_i^{new} SK^*)$ Checks $V_4^* \stackrel{?}{=} V_4$ Replaces $\{PID_i\}$ to $\{PID_i^{new}\}$ in the smart card.	Computes $N_3^* = M_4 \oplus h(KS_j N_2)$ $SK^* = h(h(N_2 HID_i) N_3^* N_1)$ $V_3^* = h(SK^* N_3^* SID_j)$ Checks $V_3^* \stackrel{?}{=} V_3$ Computes $x^{new} = h(x N_2)$ $PID_i^{new} = HID_i \oplus h(x^{new} k_{GWN})$ $P_i = PID_i^{new} \oplus h(N_1 HID_i)$ $M_5 = N_2 \oplus h(HID_i SID_j N_1)$ $M_6 = N_3 \oplus h(N_2 HID_i PID_i^{new})$ $V_4 = h(N_2 N_3 PID_i^{new} SK)$ If the key agreement is successful, updates $\{PID_i, x\}$ to $\{PID_i^{new}, x^{new}\}$. $\{P_i, M_5, M_6, V_4\}$	

Figure 7. Login and authentication phase of WSN-SLAP.

Step 1: After inserting the smart card, U_i inputs the identity ID_i and the password PW_i . The smart card computes $R_1^* = SR_i \oplus h(ID_i || PW_i)$, $APW_i^* = h(PW_i || R_i)$ and $V_i^* = h(APW_i^* || ID_i || R_1^*)$. Then, the smart card checks the validity of V_i^* compared with V_i stored in the smart card. If the validity is confirmed, the smart card generates a random nonce N_1 , and computes $HID_i = SHID_i \oplus h(PW_i || ID_i || R_i)$, $S_i = SID_j \oplus h(PID_i || HID_i)$, $M_1 = N_1 \oplus h(HID_i || PID_i)$, and $V_1 = h(SID_j || PID_i || N_1 || HID_i)$. At last, U_i sends $\{PID_i, S_i, M_1, V_1\}$ to GW over a public channel.

Step 2: When GW receives $\{PID_i, S_i, M_1, V_1\}$ from U_i , GW retrieves PID_i and the shared secret value x from GW's database. Then, GW computes $HID_i^* = PID_i \oplus h(x || k_{GWN})$, $SID_j^* = S_i \oplus h(PID_i || HID_i^*)$, $N_1^* = M_1 \oplus h(HID_i^* || PID_i)$ and $V_1^* = h(SID_j^* || PID_i || N_1^* || HID_i^*)$, and checks the validity of V_1^* compared with V_1 . If the validity is confirmed, GW retrieves SID_j and $h(SID_j || R_j)$ from GW's database. GW computes $KS_j = h(h(SID_j || R_j) || k_{GWN})$, $M_2 = h(N_2 || HID_i) \oplus h(KS_j || PID_i)$, $M_3 = N_1 \oplus h(h(N_2 || HID_i) || KS_j)$, and $V_2 = h(PID_i || SID_j || h(N_2 || HID_i) || N_1)$. At last, GW sends $\{PID_i, M_2, M_3, V_2\}$ to S_j over a public channel.

Step 3: If S_j receives $\{PID_i, M_2, M_3, V_2\}$, S_j computes $h(N_2 || HID_i)^* = M_2 \oplus h(KS_j || PID_i)$, $N_1^* = M_3 \oplus h(h(N_2 || HID_i)^* || PID_i)$, $V_2^* = h(PID_i || SID_j || h(N_2 || HID_i) || N_1^*)$ and checks the validity of V_2^* compared with the parameter V_2 . If the validity is confirmed, S_j computes $SK = h(h(N_2 || HID_i) || N_3 || N_1)$, $M_4 = N_3 \oplus h(KS_j || N_2)$, $V_3 = h(SK || N_3 || SID_j)$, where SK is a session key. Finally, S_j sends $\{M_4, V_3\}$ to GW.

Step 4: After receiving the message $\{M_4, V_3\}$ from S_j , GW computes $N_3^* = M_4 \oplus h(KS_j || N_2)$, $SK^* = h(h(N_2 || HID_i) || N_3^* || N_1)$, $V_3^* = h(SK^* || N_3^* || SID_j)$ and verifies the equality of V_3^* and V_3 . If the verification is successful, GW generates a random nonce N_2

and computes $x^{new} = h(x||N_2)$, $PID_i^{new} = HID_i \oplus h(x^{new}||k_{GWN})$, $P_i = PID_i^{new} \oplus h(N_1||HID_i)$, $M_5 = N_2 \oplus h(HID_i||SID_j||N_1)$, $M_6 = N_3 \oplus h(N_2||HID_i||PID_i^{new})$ and $V_4 = h(N_2||N_3||PID_i^{new}||SK)$. At last, GW sends $\{P_i, M_5, M_6, V_4\}$ to U_i and updates $\{PID_i, x\}$ to $\{PID_i^{new}, x^{new}\}$ if the key agreement is successful.

Step 5: When U_i receives the message $\{P_i, M_5, M_6, V_4\}$ from GW, U_i computes $PID_i^{new} = P_i \oplus h(N_1||HID_i)$, $N_2^* = M_5 \oplus h(HID_i||SID_j||N_1)$, $N_3^* = M_6 \oplus h(N_2||HID_i||PID_i^{new})$, $SK^* = h(h(N_2^*||HID_i)||N_3^*||N_1)$, $V_4^* = h(N_2^*||N_3^*||PID_i^{new}||SK^*)$ and checks the validity of V_4^* compared with V_4 . If the validity is confirmed, U_i replaces $\{PID_i\}$ to $\{PID_i^{new}\}$ in the smart card.

5.4. Password Update Phase

In WSN-SLAP, users can easily change their own password. The details are shown as below.

Step 1: After inserting the smart card, The user U_i inputs the identity ID_i and the password PW_i . The smart card computes $R_i^* = SR_i \oplus h(ID_i||PW_i)$, $APW_i^* = h(PW_i||R_i)$, $V_i^* = h(APW_i||ID_i||R_i^*)$ and verifies the equality of V_i^* and V_i . If the verification is successful, the smart card requests a new password to U_i .

Step 2: U_i inputs a new password PW_i^{new} . The smart card selects a random number R_i^{new} and computes $APW_i^{new} = h(PW_i^{new}||R_i^{new})$, $SR_i^{new} = R_i^{new} \oplus (ID_i||PW_i^{new})$, $SHID_i^{new} = HID_i \oplus h(PW_i^{new}||ID_i||R_i^{new})$, $V_i^{new} = h(APW_i^{new}||ID_i||R_i^{new})$. Finally, the smart card stores $\{SR_i^{new}, SHID_i^{new}, V_i^{new}, PID_i, h(\cdot)\}$.

5.5. Sensor Node Addition Phase

To add a new sensor node S_j^{new} to WSN-SLAP, S_j^{new} registers to the gateway GW. The detailed steps are described as follows.

Step 1: S_j^{new} selects its identity SID_j^{new} . Then, S_j^{new} generates a random number R_j^{new} . With SID_j^{new} and R_j^{new} , S_j^{new} computes $h(SID_j^{new}||R_j^{new})$ and sends $\{SID_j^{new}, h(SID_j^{new}||R_j^{new})\}$ to GW through a secure channel.

Step 2: After receiving $\{SID_j^{new}, h(SID_j^{new}||R_j^{new})\}$ from S_j^{new} , GW computes $KS_j^{new} = h(h(SID_j^{new}||R_j^{new})||k_{GWN})$ and stores $\{SID_j^{new}, h(SID_j^{new}||R_j^{new})\}$ in the database of GW. Finally, GW sends $\{KS_j^{new}\}$ to S_j^{new} .

Step 3: S_j^{new} receives the message $\{KS_j^{new}\}$ from GW and stores $\{KS_j^{new}\}$ in the memory of S_j^{new} .

6. Security Analysis

WSN-SLAP not only considers lightweight features using hash functions and XOR operations, but also ensures a higher security level compared with related schemes. To evaluate the security of WSN-SLAP, we perform informal security analysis and formal security analysis such as BAN logic, ROR model, and AVISPA simulation tool. We show that WSN-SLAP prevents a variety of attacks using informal analysis. We demonstrate the mutual authentication of WSN-SLAP using BAN logic and also prove the session key security of WSN-SLAP by using the ROR model. We use the AVISPA simulation tool to prove security features of WSN-SLAP against replay and MITM attacks.

6.1. Informal Security Analysis

WSN-SLAP provides security against various attacks such as insider, stolen smart card, replay, sensor node capture, off-line password guessing, privileged insider, stolen verifier, and MITM attacks. Furthermore, WSN-SLAP ensures perfect forward secrecy and mutual authentication.

6.1.1. Insider Attack

If an adversary \mathcal{A} registers to the gateway GW as a legal user, \mathcal{A} can authenticate to GW and the sensor node S_j . \mathcal{A} captures messages $\{PID_i, M_2, M_3, V_2\}$, $\{M_4, V_3\}$ and $\{P_i, M_5, M_6, V_4\}$. Then, \mathcal{A} computes $h(h(N_2||HID_i)||KS_j) = M_3 \oplus N_1$ and $h(KS_j||PID_i) = M_2 \oplus h(N_2||HID_i)$. To compromise other legal user's sessions, \mathcal{A} must need KS_j to compute the session key. Since hash functions mask the random nonce N_2 and the user's secret parameter HID_i such as $h(h(N_2||HID_i)||KS_j)$, \mathcal{A} cannot compute the shared secret parameter KS_j between GW and S_j . Therefore, WSN-SLAP is secure against the insider attacks.

6.1.2. Stolen Smart Card Attack

Suppose that an adversary \mathcal{A} captures the legal user U_i 's smart card. Then, \mathcal{A} uses the power analysis attack to extract stored parameters in the smart card. With U_i 's smart card parameters, \mathcal{A} tries to authenticate with the gateway GW and the sensor node S_j . However, \mathcal{A} cannot compute the login request message $\{PID_i, S_i, M_1, V_1\}$ because HID_i is masked by $SHID_i = HID_i \oplus h(PW_i||ID_i||R_i)$. To calculate HID_i , \mathcal{A} needs to guess ID_i and PW_i at the same time. Since these tasks are computationally infeasible task, it is hard to obtain both ID_i and PW_i . For these reasons, WSN-SLAP is secure against stolen smart card attacks.

6.1.3. Replay Attack

If an adversary \mathcal{A} intercepts messages $\{PID_i, M_2, M_3, V_2\}$ and $\{ID_i, S_i, M_1, V_1\}$ from a legal user U_i , \mathcal{A} tries to authenticate with the gateway GW by sending intercepted messages at other sessions. In WSN-SLAP, GW and the sensor node check the freshness of random nonces N_1, N_2 and N_3 . Thus, WSN-SLAP can provide security against replay attacks.

6.1.4. Sensor Node Capture Attack

We assume that an adversary \mathcal{A} captures a specific sensor node S_j and obtains parameters $\{SID_j, KS_j\}$ from the S_j 's memory by using the power analysis attack. Then, \mathcal{A} can authenticate with gateway GW and user U_i . However, \mathcal{A} cannot threat other sensor nodes. Since the shared secret parameter $KS_j = h(h(SID_j||R_j)||k_{GWN})$, \mathcal{A} can only authenticate with the specific sensor node S_j . \mathcal{A} cannot calculate any information about other sensor nodes. Therefore, WSN-SLAP is secure against sensor node capture attacks.

6.1.5. Off-Line Password Guessing Attack

According to Section 1.2, an adversary \mathcal{A} can guess a legal user U_i 's password PW_i . \mathcal{A} can also extract stored parameters $\{SR_i, SHID_i, V_i, PID_i, h(\cdot)\}$ from U_i 's legitimate smart card. Then, \mathcal{A} tries to impersonate as U_i . However, \mathcal{A} cannot compute $R_i = SR_i \oplus h(ID_i||PW_i)$ to obtain $HID_i = SHID_i \oplus h(PW_i||ID_i||R_i)$ without knowing the identity ID_i . Therefore, \mathcal{A} cannot compute the legal message $\{PID_i, M_2, M_3, V_2\}$. Accordingly, WSN-SLAP has resistance to off-line password-guessing attacks.

6.1.6. Privileged Insider Attack

If a privileged insider adversary \mathcal{A} intercepts a legal user U_i 's registration message $\{ID_i\}$, \mathcal{A} tries to compute U_i 's session key by using messages in Section 5.3. However, \mathcal{A} cannot compute the session key of U_i . To compute $SK = h(h(N_2||HID_i)||N_3||N_1)$, \mathcal{A} has to calculate HID_i which is the shared secret parameter between U_i and the gateway GW . However, \mathcal{A} cannot compute $HID_i = SHID_i \oplus h(PW_i||ID_i||R_i)$ from the login request message $\{PID_i, S_i, M_1, V_1\}$ without U_i 's password and the random number R_i . Consequently, WSN-SLAP ensures security against privileged insider attacks.

6.1.7. Stolen Verifier Attack

Assuming that an adversary \mathcal{A} steals the gateway GW 's verification table including $\{SID_j, h(SID_j||R_j)\}$ and (PID_i, x) . However, \mathcal{A} cannot compute the session key of the

legal user U_i with these parameters. To compute the session key $SK = h(h(N_2||HID_i)||N_3||N_1)$, \mathcal{A} must compute HID_i by using $PID_i = HID_i \oplus h(x||k_{GWN})$. Since the parameter k_{GWN} is GW 's master key, \mathcal{A} cannot compute HID_i . Therefore, WSN-SLAP has resistance to stolen verifier attacks.

6.1.8. MITM Attack

During the login and authentication phase, an adversary \mathcal{A} intercepts and tries to modify the login request message $\{PID_i, S_i, M_1, V_1\}$. However, the gateway GW can easily detect the modified message by using the verification table. In addition, it is impossible to modify all messages because they include random parameters. Therefore, WSN-SLAP can prevent MITM attacks.

6.1.9. Session-Specific Random Number Leakage Attack

Assume that an adversary \mathcal{A} obtains all random parameters N_1, N_2 , and N_3 . Then, \mathcal{A} tries to compute the session key SK . However, it is impossible to calculate the session key without knowing HID_i . HID_i is masked with the secret key x and the master key k_{GWN} during the session. Accordingly, WSN-SLAP is secure against session-specific random number leakage attacks.

6.1.10. Perfect Forward Secrecy

We suppose that an adversary \mathcal{A} obtains GW 's master key k_{GWN} . Then, \mathcal{A} tries to compute the session key $SK = h(h(N_2||HID_i)||N_3||N_1)$ of the user U_i . However, the master key k_{GWN} is utilized, i.e., $h(x||k_{GWN})$ and $h(h(SID_j||R_j)||k_{GWN})$. Therefore, \mathcal{A} needs the shared secret parameter x or $h(SID_j||R_j)$ to analyze the secret parameter. For this reason, WSN-SLAP provides perfect forward secrecy.

6.1.11. Mutual Authentication

To authenticate with each other, each participant of WSN-SLAP performs verification processes. The gateway GW checks the validity of $V_1 \stackrel{?}{=} V_1^*$ and $V_3 \stackrel{?}{=} V_3^*$, the sensor node S_j verifies $V_2 \stackrel{?}{=} V_2^*$, and the U_i checks $V_4 \stackrel{?}{=} V_4^*$. If the whole verification process is successful, we can conclude that each participant is authenticated with each other. Therefore, WSN-SLAP guarantees mutual authentication.

6.2. BAN Logic

In this section, we prove mutual authentication of WSN-SLAP using BAN logic analysis [8]. BAN logic has been widely used to analyze the mutual authentication of various authentication schemes [32,33]. In WSN-SLAP, the participants authenticate with each other to establish a session key SK among U , GW , and SN . Table 2 presents the basic notations of the BAN logic used in this proof.

Table 2. The basic notations.

Notation	Description
P_1, P_2	Two principals
S_1, S_2	Two statements
SK	The session key
$P_1 \equiv S_1$	P_1 believes S_1
$P_1 \sim S_1$	P_1 once said S_1
$P_1 \Rightarrow S_1$	P_1 controls S_1
$P_1 \triangleleft S_1$	P_1 receives S_1
$\#S_1$	S_1 is fresh
$\{S_1\}_{Key}$	S_1 is encrypted with Key
$P_1 \xleftrightarrow{Key} P_2$	P_1 and P_2 have shared key Key

6.2.1. Rules

The logical rules of the BAN logic are described as below.

1. Message meaning rule (MMR) :

$$\frac{P_1 \mid \equiv P_1 \xleftrightarrow{Key} P_2, \quad P_1 \triangleleft (S_1)_{Key}}{P_1 \mid \equiv P_2 \mid \sim S_1}$$

2. Nonce verification rule (NVR) :

$$\frac{P_1 \mid \equiv \#(S_1), \quad P_1 \mid \equiv P_2 \mid \sim S_1}{P_1 \mid \equiv P_2 \mid \equiv S_1}$$

3. Jurisdiction rule (JR) :

$$\frac{P_1 \mid \equiv P_2 \mid \implies S_1, \quad P_1 \mid \equiv P_2 \mid \equiv S_1}{P_1 \mid \equiv S_1}$$

4. Belief rule (BR) :

$$\frac{P_1 \mid \equiv (S_1, S_2)}{P_1 \mid \equiv S_1}$$

5. Freshness rule (FR) :

$$\frac{P_1 \mid \equiv \#(S_1)}{P_1 \mid \equiv \#(S_1, S_2)}$$

6.2.2. Goals

In WSN-SLAP, the basic goals of the BAN logic are that each principal establishes a session key and achieves mutual authentication. The goals for proving mutual authentication of WSN-SLAP are defined as follows :

Goal 1: $U \mid \equiv U \xleftrightarrow{SK} GW$

Goal 2: $U \mid \equiv GW \mid \equiv U \xleftrightarrow{SK} GW$

Goal 3: $GW \mid \equiv U \xleftrightarrow{SK} GW$

Goal 4: $GW \mid \equiv U \mid \equiv U \xleftrightarrow{SK} GW$

Goal 5: $SN \mid \equiv SN \xleftrightarrow{SK} GW$

Goal 6: $SN \mid \equiv GW \mid \equiv SN \xleftrightarrow{SK} GW$

Goal 7: $GW \mid \equiv SN \xleftrightarrow{SK} GW$

Goal 8: $GW \mid \equiv SN \mid \equiv SN \xleftrightarrow{SK} GW$

6.2.3. Idealized Forms

In WSN-SLAP, the authentication request and response messages $\{PID_i, S_i, M_1, V_1\}$, $\{PID_i, M_2, M_3, V_2\}$, $\{M_4, V_3\}$, and $\{P_i, M_5, M_6, V_4\}$ are transmitted through a public channel. We will transmit these messages into the idealized form and omit other messages because they cannot efficiently provide the logical properties of BAN logic. WSN-SLAP's idealized form messages are shown as below:

$$Msg_1 : U \rightarrow GW : \{N_1, SID_j\}_{HID_i}$$

$$Msg_2 : GW \rightarrow SN : \{h(N_2 || HID_i), N_1\}_{KS_j}$$

$$Msg_3 : SN \rightarrow GW : \{N_3\}_{KS_j}$$

$$Msg_4 : GW \rightarrow U : \{N_2, N_3\}_{HID_1}$$

6.2.4. Assumptions

After the registration phase, each principal believes that it has secret keys which are shared among each other. The principal also trusts that random numbers and pseudo identity are fresh. Moreover, the principal believes that a legal principal can control the entitled components and values. The assumptions of the BAN logic in WSN-SLAP are as below:

$$A_1: GW | \equiv \#(N_1)$$

$$A_2: GW | \equiv \#(N_3)$$

$$A_3: SN | \equiv \#(h(N_2 || HID_i))$$

$$A_4: U | \equiv \#(N_2)$$

$$A_5: U | \equiv GW \Rightarrow (U \xleftrightarrow{SK} GW)$$

$$A_6: GW | \equiv U \Rightarrow (U \xleftrightarrow{SK} GW)$$

$$A_7: SN | \equiv GW \Rightarrow (SN \xleftrightarrow{SK} GW)$$

$$A_8: GW | \equiv SN \Rightarrow (SN \xleftrightarrow{SK} GW)$$

$$A_9: U | \equiv U \xleftrightarrow{HID_i} GW$$

$$A_{10}: GW | \equiv U \xleftrightarrow{HID_i} GW$$

$$A_{11}: SN | \equiv SN \xleftrightarrow{KS_j} GW$$

$$A_{12}: GW | \equiv SN \xleftrightarrow{KS_j} GW$$

6.2.5. BAN Logic Proof

We conduct the BAN logic analysis of WSN-SLAP as follows:

Step 1: S_1 can be obtained from Msg_1 .

$$S_1 : GW \triangleleft \{N_1, SID_j\}_{HID_i}$$

Step 2: S_2 can be induced by applying the MMR using S_1 and A_{10} .

$$S_2 : GW | \equiv U | \sim (N_1, SID_j)$$

Step 3: S_3 can be induced by applying the FR using S_2 and A_1 .

$$S_3 : GW | \equiv \#(N_1, SID_j)$$

Step 4: S_4 can be induced by applying the NVR using S_2 and S_3 .

$$S_4 : GW | \equiv U | \equiv (N_1, SID_j)$$

Step 5: S_5 is can be induced by S_4 and the BR.

$$S_5 : GW | \equiv U | \equiv (N_1)$$

Step 6: S_6 is obtained from Msg_2 .

$$S_6 : SN \triangleleft \{h(N_2 || HID_i), N_1\}_{KS_j}$$

Step 7: S_7 is can be induced by applying the MMR using S_6 and A_{13} .

$$S_7 : SN | \equiv GW | \sim (h(N_2 || HID_i), N_1)$$

Step 8: S_8 is can be induced by applying the FR using S_7 and A_3 .

$$S_8 : SN | \equiv \#(h(N_2 || HID_i), N_1)$$

Step 9: S_9 is can be induced by applying the NVR using S_7 and S_8 .

$$S_9 : SN | \equiv GW | \equiv (h(N_2 || HID_i), N_1)$$

Step 10: S_{10} is obtained from Msg_3 .

$$S_{10} : GW \triangleleft \{N_3\}_{KS_j}$$

Step 11: S_{11} can be induced by applying the MMR using A_5 and S_8 .

$$S_{11} : GW | \equiv SN | \sim (N_3)$$

Step 12: S_{12} can be induced by applying the NVR using S_9 and S_{10} .

$$S_{12} : GW | \equiv SN | \equiv (N_3)$$

Step 13: S_{13} and S_{14} can be induced by S_9 , and S_{12} . SN and GW can compute the session key $SK = h(h(N_2 || HID_i) || N_3 || N_1)$.

$$S_{13} : GW | \equiv SN | \equiv (SN \xleftrightarrow{SK} GW) \quad (\text{Goal 8})$$

$$S_{14} : SN | \equiv GW | \equiv (SN \xleftrightarrow{SK} GW) \quad (\text{Goal 6})$$

Step 14: S_{15} and S_{16} can be induced by applying the JR using S_{13} and A_8 , and S_{14} and A_7 , respectively.

$$S_{15} : GW | \equiv (SN \xleftrightarrow{SK} GW) \quad (\text{Goal 7})$$

$$S_{16} : SN | \equiv (SN \xleftrightarrow{SK} GW) \quad (\text{Goal 5})$$

Step 15: S_{17} is obtained from Msg_4 .

$$S_{17} : U \triangleleft \{N_2, N_3\}_{HID_i}$$

Step 16: S_{18} can be induced by A_9 , S_{17} , and the MMR.

$$S_{18} : U | \equiv GW | \sim (N_2, N_3)$$

Step 17: S_{19} can be induced by applying the FR using S_{18} and A_4 .

$$S_{19} : U | \equiv \#(N_2, N_3)$$

Step 18: S_{20} can be induced by S_{16} , S_{17} , and the NVR.

$$S_{20} : U | \equiv GW | \equiv (N_2, N_3)$$

Step 19: S_{21} and S_{22} can be induced by S_5 , S_{18} . U and GW can compute the session key $SK = h(h(N_2 || HID_i) || N_3 || N_1)$

$$S_{21} : U | \equiv GW | \equiv (U \xleftrightarrow{SK} GW) \quad (\text{Goal 2})$$

$$S_{22} : GW | \equiv U | \equiv (U \xleftrightarrow{SK} GW) \quad (\text{Goal 4})$$

Step 20: S_{23} and S_{24} can be induced by applying the JR using S_{21} and A_5 , S_{22} , and A_6 , respectively.

$$S_{23} : U | \equiv (U \xleftrightarrow{SK} GW) \quad (\text{Goal 1})$$

$$S_{24} : GW | \equiv (U \xleftrightarrow{SK} GW) \quad (\text{Goal 3})$$

6.3. ROR Model

This section proves the security of the session key of WSN-SLAP by using the well-known Real-Or-Random (ROR) model [9]. In WSN-SLAP, there are three participants. $\mathcal{P}_U^{t_1}$ is a user, $\mathcal{P}_{GW}^{t_2}$ is a gateway, and $\mathcal{P}_{GW}^{t_3}$ is a sensor node. In the ROR model, the network is under an adversary \mathcal{A} who can eavesdrop, capture, insert, and delete messages. With these abilities, \mathcal{A} performs various attacks using *Execute*, *CorruptSC*, *Reveal*, *Send*, and *Test* queries.

- *Execute* : This query is a passive attack that \mathcal{A} can eavesdrop the legal entity's message.
- *CorruptSC* : This query means \mathcal{A} obtains stored parameters from the user's smart card.
- *Reveal* : This query means \mathcal{A} reveals the session key SK .
- *Send* : This query is an active attack that \mathcal{A} sends a message to receive a response message.
- *Test* : An adversary \mathcal{A} obtains a flipped unbiased coin before the game starts. If \mathcal{A} obtains $c = 1$, it means the session key SK is fresh. If \mathcal{A} obtains $c = 0$, it means the session key is not fresh. Otherwise, \mathcal{A} obtains a *NULL* value. To ensure the security of the session key, it is necessary that \mathcal{A} cannot distinguish the result value between a random number and the session key.

Security Proof

Theorem 1. Let \mathcal{A} attempt to obtain the session key of WSN-SLAP in polynomial time as follows. $Adv_{\mathcal{A}}(\text{Poly})$ is the probability of the session key being broken by \mathcal{A} . q_h^2 , HASH, and q_{send} mean the number of hash queries, the range space of the hash function, and the number of send queries, respectively. s' and C' are the Zipf's parameters [34].

$$Adv_{\mathcal{A}}(\text{Poly}) \leq \frac{q_h^2}{|\text{HASH}|} + 2\{C'q_{\text{send}}^{s'}\}$$

We follow the proof according to the method of [35,36]. We perform four games Game_k , where $k \in [0, 3]$. $\text{Succ}_{\mathcal{A}, \text{Game}_k}$ is the event that \mathcal{A} can guess a correct bit c in the Game_k , and $\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_k}]$ is the probability of $\text{Succ}_{\mathcal{A}, \text{Game}_k}$. We can perform Game_k as follows with these parameters.

- Game_0 : This game describes a real attack of \mathcal{A} in WSN-SLAP under the ROR model. The random bit c needs to be selected before starting the game. Therefore, we can derive as follows.

$$Adv_{\mathcal{A}}(\text{Poly}) = |2\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_0}] - 1| \quad (1)$$

- Game_1 : In the Game_1 , \mathcal{A} obtains each entity's messages $\{PID_i, S_i, M_1, V_1\}$, $\{PID_i, M_2, M_3, V_2\}$, $\{M_4, V_3\}$, and $\{P_i, M_5, M_6, V_4\}$ using Execute query. Then, \mathcal{A} performs Test and Reveal queries to obtain the session key SK. Since $SK = h(h(N_2||HID_i)||N_3||N_1)$, \mathcal{A} has to get random nonces N_1 , N_2 , and N_3 . In addition, \mathcal{A} needs the user's masked identity HID_i . For these reasons, \mathcal{A} cannot calculate SK. This means Game_0 and Game_1 are indistinguishable. Therefore, we can get the following equivalent.

$$\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_1}] = \Pr[\text{Succ}_{\mathcal{A}, \text{Game}_0}] \quad (2)$$

- Game_2 : In this game, \mathcal{A} performs Send query, which is an active attack. \mathcal{A} utilizes $\{PID_i, S_i, M_1, V_1\}$, $\{PID_i, M_2, M_3, V_2\}$, $\{M_4, V_3\}$, and $\{P_i, M_5, M_6, V_4\}$ to get the session key SK. Parameters V_1 , V_2 , V_3 , and V_4 are masked by HASH query. In addition, parameters PID_i , M_1 , M_2 , M_3 , M_4 , M_5 , M_6 , and P_i contain random nonces N_1 , N_2 , and N_3 . By using random nonces, we can prevent collision from other sessions. According to the birthday paradox [37], we can get the following inequation.

$$|\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_2}] - \Pr[\text{Succ}_{\mathcal{A}, \text{Game}_1}]| \leq \frac{q_h^2}{|\text{HASH}|} \quad (3)$$

- Game_3 : In the Game_3 , \mathcal{A} executes CorruptSC query and obtains smart card's stored parameters $\{SR_i, SHID_i, V_i, PID_i\}$ by using the power analysis attack, where $SR_i = R_i \oplus h(ID_i||PW_i)$, $SHID_i = HID_i \oplus h(PW_i||ID_i||R_i)$, $V_i = h(APW_i||ID_i||R_i)$, and $PID_i = HID_i \oplus h(x||k_{GWN})$. To obtain R_i and HID_i , \mathcal{A} needs the identity ID_i and the password PW_i . Therefore, \mathcal{A} cannot distinguish with Game_2 and Game_3 if guessing PW_i is computationally infeasible task. Then, we can obtain the result by using Zipf's law [34].

$$|\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_3}] - \Pr[\text{Succ}_{\mathcal{A}, \text{Game}_2}]| \leq C'q_{\text{send}}^{s'} \quad (4)$$

Finally, \mathcal{A} gets the guessed bit c because games are done.

$$\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_3}] = \frac{1}{2} \quad (5)$$

Moreover, we can get the following result by using (1) and (2).

$$\frac{1}{2}Adv_{\mathcal{A}}(\text{Poly}) = |\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_0}] - \frac{1}{2}| = |\Pr[\text{Succ}_{\mathcal{A}, \text{Game}_1}] - \frac{1}{2}| \quad (6)$$

Using (5) and (6), we obtain the following equation.

$$\frac{1}{2}Adv_{\mathcal{A}}(Poly) = |Pr[Succ_{\mathcal{A},Game_1}] - Pr[Succ_{\mathcal{A},Game_3}]| \quad (7)$$

We get the following result utilizing the triangular inequality.

$$\begin{aligned} \frac{1}{2}Adv_{\mathcal{A}}(Poly) &= |Pr[Succ_{\mathcal{A},Game_1}] - Pr[Succ_{\mathcal{A},Game_3}]| \\ &\leq |Pr[Succ_{\mathcal{A},Game_1}] - Pr[Succ_{\mathcal{A},Game_2}]| \\ &\quad + |Pr[Succ_{\mathcal{A},Game_2}] - Pr[Succ_{\mathcal{A},Game_3}]| \\ &\leq \frac{q_h^2}{2|HASH|} + C'q_{send}^{s'} \end{aligned} \quad (8)$$

By multiplying (8) by 2, we get the following result.

$$Adv_{\mathcal{A}}(Poly) \leq \frac{q_h^2}{|HASH|} + 2\{C'q_{send}^{s'}\}$$

Therefore, we prove

6.4. AVISPA Simulation

In this section, we analyze security features of WSN-SLAP by using AVISPA [10,11]. AVISPA is a formal security verification tool that detects MITM and replay attacks against the authentication protocol.

AVISPA uses the High-Level Protocols Specification Language (HLPSL). After receiving a protocol written in HLPSL, the translator converts the HLPSL-based protocol to an intermediate format (IF). Then, the translator inputs the IF to four back-ends, which are Constraint Logic-based Attack Searcher (CL-AtSe), Tree Automata based on Automatic Approximations for Analysis of Security Protocol (TA4SP), SAT-based Model-Checker (SATMC), and On the fly Model-Checker (OFMC), respectively. Consequently, the IF is converted to an output format (OF). If the summary of OF is SAFE, it means the protocol has resistance to replay and MITM attacks.

Specifically, OFMC back-end can utilize XOR operations. Therefore, we use this back-end in our paper.

6.4.1. HLPSL Specifications

In HLPSL, WSN-SLAP consists of users UA , gateway GWN , and sensor nodes SN . These entities are written as *role*. There are also two *composition roles* named *session* and *environment*, which contain security goals. Figure 8 indicates goals and the *role* of *session* and *environment* of WSN-SLAP.

Figure 9 shows the whole process of the user UA . In state 1, the user UA registers to GWN . To start the session, UA receives the *start* message. Then, UA sends a registration request message $\{ID_i\}$ to the gateway GWN through a secure channel. In state 2, UA receives a smart card from GWN and stores $\{R_i, SR_i, SHID_i, V_i\}$ in the smart card. In the login and authentication phase, UA sends $\{PID_i, S_i, M_1, V_1\}$ to GWN via a public channel. The function $witness(UA, GWN, ua_gw_n1, N_1')$ indicates the freshness of N_1 generated by UA . In State 3, UA receives $\{P_i, M_5, M_6, V_4\}$ from GWN . Then, UA authenticates with GWN using N_2' in $request(GWN, UA, gw_ua_n3, N_2')$.

```

role session(UA, SN, GWN : agent, SKuagwn, SKsngwn : symmetric_key, H: hash_func)
def=
local SN1, SN2, SN3, RV1, RV2, RV3: channel(dy)
composition
user(UA, SN, GWN, SKuagwn, SKsngwn, H, SN1, RV1)
^ sen(UA, SN, GWN, SKuagwn, SKsngwn, H, SN2, RV2)
^ gate(UA, SN, GWN, SKuagwn, SKsngwn, H, SN3, RV3)
end role
role environment()
def=
const ua, sn, gwn : agent,
skuagwn, sksngwn: symmetric_key,
h: hash_func,
idi, pidi, sidj: text,
ua_gw_n1, gw_sn_n2, sn_gw_n3, gw_ua_n3: protocol_id,
sp1, sp2, sp3, sp4, sp5, sp6: protocol_id
intruder_knowledge = {idi, pidi, sidj, h}
composition
session(ua, sn, gwn, skuagwn, sksngwn, h)^/session(i, sn, gwn, skuagwn, sksngwn, h)
^/session(ua, i, gwn, skuagwn, sksngwn, h)
^/session(ua, sn, i, skuagwn, sksngwn, h)
end role
goal
secrecy_of sp1, sp2, sp3, sp4, sp5, sp6
authentication_on ua_gw_n1
authentication_on gw_sn_n2
authentication_on sn_gw_n3
authentication_on gw_ua_n3
end goal
environment()

```

Figure 8. Role of session, environment and goal.

```

role user(UA, SN, GWN: agent, SKuagwn, SKsngwn : symmetric_key, H: hash_func, SND, RCV : channel(dy))
played_by UA
def=
local State: nat,
IDi, PWi, X, Rg, PIDi, APWi, Ri, SRi, SHIDi, Vi, HIDi, Kgwn, SIDj, Rj, KSj: text,
N1, Si, M1, V1, N2, M2, M3, V2, N3, SK, M4, V3, Xnew, PIDinew, Mpid, M5, M6, V4: text

const sp1, sp2, sp3, sp4, sp5, sp6, ua_gw_n1, gw_sn_n2, sn_gw_n3, gw_ua_n3: protocol_id
init State := 0
transition

%%Registration phase
1. State = 0 ^ RCV(start) =>
State' := 1 ^ SND({IDi}_SKuagwn)
^ secret({IDi}, sp1, {UA,GWN})

%%Recieve smartcard
2. State = 1 ^ RCV ((xor(H(IDi.Rg),H(X.Kgwn)),H(IDi.Rg))_SKuagwn)=>
State' := 2 ^ Rf' := new() ^ APWi' := H(PWi.Ri)
^ SRf' := xor(Rf', H(IDi.PWi))
^ SHIDf' := xor(H(IDi.Rg'), H(PWi.IDi.Rf'))
^ Vf' := H(APWi.IDi.Rf')
^ secret({Rf',PWf'}, sp2, UA)

%%Login & Authentication phase
^ N1' := new() ^ Sf' := xor(SIDj, H(xor(H(IDi.Rg'),H(X.Kgwn)),H(IDi.Rg)))
^ M1f' := xor(N1', H(H(IDi.Rg'),xor(H(IDi.Rg'),H(X.Kgwn))))
^ V1f' := H(SIDj.xor(H(IDi.Rg'),H(X.Kgwn)),N1'.H(IDi.Rg'))
^ SND(xor(H(IDi.Rg'),H(X.Kgwn)), Sf'.M1f'.V1f')
^ witness(UA,GWN,ua_gw_n1,N1f')

3. State = 2 ^ RCV(xor(xor(H(IDi.Rg'),H(X.N2'.Kgwn)), H(N1'.H(IDi.Rg'))).xor(N2', H(H(IDi.Rg').SIDj.N1f')).xor(N3',
H(N2'.H(IDi.Rg').xor(H(IDi.Rg'),H(X.N2'.Kgwn))))).H(N2'.N3'.xor(H(IDi.Rg'),H(X.N2'.Kgwn)).H(H(N2'.H(IDi.Rg')).N3'.N1f')) =>
State' := 3 ^ request(GWN,UA,gw_ua_n3, N2f')
end role

```

Figure 9. Role of user.

6.4.2. Simulation Result

If the protocol's result summary is SAFE in OFMC simulation, the protocol has resistance to replay and MITM attacks. The result of WSN-SLAP's AVISPA simulation tool using OFMC back-end is shown in Figure 10. Thus, WSN-SLAP can prevent replay and MITM attacks.

```

% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/span/span/testsuite/results/dk.if
GOAL
as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 26.60s
visitedNodes: 2704 nodes
depth: 12 plies

```

Figure 10. Result of the Automated Verification of Internet Security Protocols and Applications (AVISPA) simulation.

7. Performance Analysis

In this section, we estimate computational costs, communication costs, and security properties of WSN-SLAP compared with existing related schemes [6,27,28,31].

7.1. Computational Costs

We analyze WSN-SLAP's computational cost compared with the performance of the related schemes [6,27,28,31]. According to [6,38], the execution time of each operation is acquired on a computer with a four-core 3.2 GHz CPU, and 8 GB memory. We estimate that T_h , T_{ecm} , and T_{sym} are the execution time of the hash function (≈ 0.00032 s), ECC point multiplication (≈ 0.0171 s), and symmetric encryption/decryption (≈ 0.0056 s), respectively. We do not consider the execution time of the XOR operation because it is negligible. Table 3 indicates the result for computational costs. Accordingly, WSN-SLAP has a more efficient computational cost than related schemes [6,27,28,31].

Table 3. Computational costs comparison.

Schemes	User	Gateway	Sensor Node	Total	Total Cost (s)
Choi et al. [27]	$9T_h + 3T_{ecm}$	$6T_h + 2T_{ecm}$	$5T_h + 1T_{ecm}$	$20T_h + 6T_{ecm}$	0.109
Wu et al. [28]	$12T_h + 2T_{ecm} + 1T_{sym}$	$11T_h + 2T_{sym}$	$4T_h + 2T_{ecm} + 1T_{sym}$	$27T_h + 4T_{ecm} + 4T_{sym}$	0.09944
Wu et al. [31]	$13T_h + 2T_{ecm}$	$13T_h$	$4T_h + 2T_{ecm}$	$30T_h + 4T_{ecm}$	0.078
Moghadam et al. [6]	$5T_h + 3T_{ecm} + 2T_{sym}$	$5T_h + 3T_{ecm} + 2T_{sym}$	$3T_h + 2T_{ecm}$	$13T_h + 8T_{ecm} + 4T_{sym}$	0.16336
Ours	$13T_h$	$18T_h$	$6T_h$	$37T_h$	0.01184

7.2. Communication Costs

We evaluate the communication cost of WSN-SLAP compared with related schemes [6,27,28,31] in this section. According to [6], we define that the user identity, sensor node identity, random number, timestamp, SHA-1 hash digest, and ECC point are 128, 16, 128, 32, 160 and 320 bits, respectively. In WSN-SLAP, the login request mes-

sage $\{PID_i, S_i, M_1, V_1\}$ requires $(160 + 160 + 160 + 160 = 640)$ bits), and the transmitted authentication messages $\{PID_i, M_2, M_3, V_2\}$, $\{M_4, V_3\}$, and $\{P_i, M_5, M_6, V_4\}$ require $(160 + 160 + 160 + 160 = 640)$ bits), $(160 + 160 = 320)$ bits), and $(160 + 160 + 160 + 160 = 640)$ bits), respectively. Consequently, total communication costs of WSL-SLAP and related schemes [6,27,28,31] are as shown in Table 4. Therefore, WSN-SLAP provides a more efficient communication cost than related schemes do [6,27,28,31].

Table 4. Communication costs comparison.

Schemes	Communication Costs	Number of Messages
Choi et al. [27]	3200 bits	4 messages
Wu et al. [28]	3296 bits	4 messages
Wu et al. [31]	3392 bits	4 messages
Moghadam et al. [6]	2512 bits	4 messages
Ours	2240 bits	4 messages

7.3. Security Properties

In Table 5, we present the security properties of WSN-SLAP with related schemes [6,27,28,31]. We show that existing protocols [6,27,28,31] suffer from various attacks, including insider, stolen smart card, and session-specific random number leakage attacks. Therefore, WSN-SLAP provides better functionality and security features compared with those of related schemes [6,27,28,31].

Table 5. Security properties.

Security Property	Choi et al. [27]	Wu et al. [28]	Wu et al. [31]	Moghadam et al. [6]	Ours
Insider Attack	○	○	×	×	○
Stolen Smart Card Attack	×	×	×	○	○
Replay Attack	○	○	○	○	○
Sensor Node Capture Attack	○	○	○	○	○
Off-line Password Guessing Attack	×	×	○	○	○
Privileged Insider Attack	○	○	×	○	○
Stolen Verifier Attack	×	○	○	○	○
MITM Attack	○	○	×	○	○
Session-Specific Random Number Leakage Attack	×	×	×	×	○
Perfect Forward Secrecy	○	○	○	×	○
Mutual Authentication	○	○	○	○	○

○: Secure from the attack. ×: Insecure from the attack.

8. Conclusions

In this paper, we discovered that Moghadam et al.'s scheme has vulnerabilities against insider, and session-specific random number leakage attacks. We also proved that Moghadam et al.'s scheme does not guarantee perfect forward secrecy. To resolve the security weaknesses of Moghadam et al.'s scheme, we proposed a secure and lightweight mutual authentication protocol for WSN environments. WSN-SLAP has resistance to various attacks, including insider, stolen smart card, off-line password guessing, stolen verifier, and session-specific random number leakage attacks. We demonstrated that WSN-SLAP provides perfect forward secrecy and mutual authentication. We proved the security of WSN-SLAP using formal security analyses, which are AVISPA, BAN logic, and ROR model. Moreover, WSN-SLAP has lightweight computational and communication costs because it

involves XOR operations and hash functions. Therefore, the proposed WSN-SLAP provides more secure and efficient communication services compared with existing related protocols and is suitable for WSN environments. In future work, we will implement a whole network and secure protocol to design a new scheme that is practical for use in WSN.

Author Contributions: Conceptualization, D.K.K.; Formal analysis, S.J.Y., J.Y.L. and S.H.S.; Investigation, S.J.Y. and Y.H.P.; Methodology, D.K.K.; Software, J.Y.L.; Supervision, Y.H.P.; Validation, J.Y.L. and S.H.S.; Writing—original draft, D.K.K.; Writing—review & editing, S.J.Y., S.H.S. and Y.H.P. All the authors contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education under Grant 2020R111A3058605, and in part by the BK21 FOUR project funded by the Ministry of Education, Korea under Grant 4199990113966.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Mandal, S.; Bera, B.; Sutrala, A.K.; Das, A.K.; Choo, K.K.R.; Park, Y. Certificateless-signcryption-based three-factor user access control scheme for IoT environment. *IEEE Internet Things J.* **2020**, *7*, 3184–3197.
- Yu, S.; Park, Y. SLUA-WSN: Secure and lightweight three-factor-based user authentication protocol for wireless sensor networks. *Sensors* **2020**, *20*, 4143. [CrossRef]
- Ghahramani, M.; Javidan, R.; Shojafar, M.; Taheri, R.; Alazab, M.; Tafazolli, R. RSS: An energy-efficient approach for securing IoT service protocols against the DoS attack. *IEEE Internet Things J.* **2020**, doi:10.1109/JIOT.2020.3023102.
- Park, K.; Noh, S.; Lee, H.; Das, A.K.; Kim, M.; Park, Y.; Wazid, M. LAKS-NVT: Provably secure and lightweight authentication and key agreement scheme without verification table in medical internet of things. *IEEE Access* **2020**, *8*, 119387–119404. [CrossRef]
- Lee, J.; Yu, S.; Park, K.; Park, Y.; Park, Y. Secure three-factor authentication protocol for multi-gateway IoT environments. *Sensors* **2019**, *19*, 2358. [CrossRef] [PubMed]
- Moghadam, M.F.; Nikooghadam, M.; Al Jabban, M.A.B.; Alishahi, M.; Mortazavi, L.; Mohajerzadeh, A. An efficient authentication and key agreement scheme based on ECDH for wireless sensor network. *IEEE Access* **2020**, *8*, 73182–73192. [CrossRef]
- Coron, J.S. Resistance against differential power analysis for elliptic curve cryptosystems. In Proceedings of the 1st International Workshop on Cryptographic Hardware and Embedded Systems, Worcester, MA, USA, 12–13 August 1999; pp. 292–302.
- Burrows, M.; Abadi, M.; Needham, R.M. A logic of authentication. *ACM Trans. Comput. Syst.* **1990**, *8*, 18–36. [CrossRef]
- Abdalla, M.; Fouque, P.; Pointcheval, D. Password-based authenticated key exchange in the three-party setting. In Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science (LNCS), Les Diablerets, Switzerland, 23–26 January 2005; pp. 65–84.
- AVISPA. Automated Validation of Internet Security Protocols and Applications. Available online: <http://www.avispa-project.org/> (accessed on 4 December 2020).
- SPAN: A Security Protocol Animator for AVISPA. Available online: <http://www.avispa-project.org/> (accessed on 4 December 2020).
- Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
- Canetti, R.; Krawczyk, H. Universally composable notions of key exchange and secure channels. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques—Advances in Cryptology (EUROCRYPT'02), Amsterdam, The Netherlands, 28 April–2 May 2002; pp. 337–351.
- Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; pp. 388–397.
- Yu, S.; Lee, J.; Lee, K.; Park, K.; Park, Y. Secure authentication protocol for wireless sensor networks in vehicular communications. *Sensors* **2018**, *18*, 3191. [CrossRef] [PubMed]
- Fu, X.; Fortino, G.; Li, W.; Pace, P.; Yang, Y. WSNs-assisted opportunistic network for low-latency message forwarding in sparse settings. *Future Gener. Comput. Syst.* **2019**, *91*, 223–237. [CrossRef]
- Fu, X.; Fortino, G.; Pace, P.; Aloï, G.; Li, W. Environment-fusion multipath routing protocol for wireless sensor networks. *Inf. Fusion* **2020**, *53*, 4–19. [CrossRef]
- Lee, J.; Yu, S.; Kim, M.; Park, Y.; Das, A.K. On the design of secure and efficient three-factor authentication protocol using honey list for wireless sensor networks. *IEEE Access* **2020**, *8*, 107046–107062. [CrossRef]
- Fu, X.; Pace, P.; Aloï, G.; Yang, L.; Fortino, G. Topology optimization against cascading failures on wireless sensor networks using a memetic algorithm. *Comput. Netw.* **2020**, *177*, 107327. [CrossRef]

20. Lamport, L. Password authentication with insecure communication. *Commun. ACM* **1981**, *24*, 770–772. [[CrossRef](#)]
21. Wong, K.H.; Zheng, Y.; Cao, J.; Wang, S. A dynamic user authentication scheme for wireless sensor networks. In Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), Taichung, Taiwan, 5–7 June 2006; pp. 1–8.
22. Tseng, H.R.; Jan, R.H.; Yang, W. An improved dynamic user authentication scheme for wireless sensor networks. In Proceedings of the IEEE Globecom, Washington, DC, USA, 26–30 November 2007; pp. 986–990.
23. Das, M.L. Two-factor user authentication in wireless sensor networks. *IEEE Trans. Wirel. Commun.* **2009**, *8*, 1086–1090. [[CrossRef](#)]
24. Khan, M.K.; Alghathbar, K. Cryptanalysis and security improvements of ‘two-factor user authentication in wireless sensor networks’. *Sensors* **2010**, *10*, 2450–2459. [[CrossRef](#)]
25. 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.
26. 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. [[CrossRef](#)]
27. Choi, Y.; Lee, D.; Kim, J.; Jung, J.; Nam, J.; Won, D. Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors* **2014**, *14*, 10081–10106. [[CrossRef](#)]
28. Wu, F.; Xu, L.; Kumari, S.; Li, X. A new and secure authentication scheme for wireless sensor networks with formal proof. *Peer-to-Peer Netw. Appl.* **2017**, *10*, 16–30. [[CrossRef](#)]
29. Nam, J.; Kim, M.; Paik, J.; Lee, Y.; Won, D. A provably-secure ECC-based authentication scheme for wireless sensor networks. *Sensors* **2014**, *14*, 21023–21044. [[CrossRef](#)] [[PubMed](#)]
30. Jiang, Q.; Ma, J.; Wei, F.; Tian, Y.; Shen, J.; Yang, Y. An untraceable temporal-credential-based two-factor authentication scheme using ECC for wireless sensor networks. *J. Netw. Comput. Appl.* **2016**, *76*, 37–48. [[CrossRef](#)]
31. Wu, F.; Xu, L.; Kumari, S.; Li, X. A privacy-preserving and provable user authentication scheme for wireless sensor networks based on Internet of Things security. *J. Ambient. Intell. Humaniz. Comput.* **2017**, *8*, 101–116. [[CrossRef](#)]
32. Ghahramani, M.; Javidan, R.; Shojafar, M. A secure biometric-based authentication protocol for global mobility networks in smart cities. *J. Supercomput.* **2020**, *76*, 8729–8755. [[CrossRef](#)]
33. Yu, S.; Lee, J.; Park, Y.; Park, Y.; Lee, S.; Chung, B. A secure and efficient three-factor authentication protocol in global mobility networks. *Appl. Sci.* **2020**, *10*, 3565. [[CrossRef](#)]
34. Wang, D.; Cheng, H.; Wang, P.; Huang, X.; Jian, G. Zipf’s law in passwords. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2776–2791. [[CrossRef](#)]
35. Wazid, M.; Bagga, P.; Das, A.K.; Shetty, S.; Rodrigues, J.J.; Park, Y. AKM-IoV: Authenticated key management protocol in fog computing-based internet of vehicles deployment. *IEEE Internet Things J.* **2019**, *6*, 8804–8817. [[CrossRef](#)]
36. Yu, S.; Lee, J.; Park, K.; Das, A.K.; Park, Y. IoV-SMAP: Secure and efficient message authentication protocol for IoV in smart city environment. *IEEE Access* **2020**, *8*, 167875–167886. [[CrossRef](#)]
37. Boyko, V.; MacKenzie, P.; Patel, S. Provably secure password-authenticated key exchange using Diffie-Hellman. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Bruges, Belgium, 14–18 May 2000; pp. 156–171.
38. Lee, C.C.; Chen, C.T.; Wu, P.H.; Chen, T.Y. Three-factor control protocol based on elliptic curve cryptosystem for universal serial bus mass storage devices. *IET Comput. Digit. Tech.* **2013**, *7*, 48–55. [[CrossRef](#)]