sensors

MDPI

Article

Secure Three-Factor Authentication Protocol for Multi-Gateway IoT Environments

JoonYoung Lee [1], SungJin Yu [1], KiSung Park [1], YoHan Park [2,*] and YoungHo Park [1,*]

1 School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea; harry250@naver.com (J.L.); darkskiln@naver.com (S.Y.); kisung2@ee.knu.ac.kr (K.P.)
2 IT Conversions, Korea Nazarene University, Cheonan, Chungcheongnam-do 31172, Korea
* Correspondence: yhpark@kornu.ac.kr (Y.P.); parkyh@knu.ac.kr (Y.P.); Tel.: +82-41-570-1629 (Y.P.); +82-53-950-7842 (Y.P.)

Received: 28 March 2019; Accepted: 20 May 2019; Published: 22 May 2019

check for updates

Abstract: Internet of Things (IoT) environments such as smart homes, smart factories, and smart buildings have become a part of our lives. The services of IoT environments are provided through wireless networks to legal users. However, the wireless network is an open channel, which is insecure to attacks from adversaries such as replay attacks, impersonation attacks, and invasions of privacy. To provide secure IoT services to users, mutual authentication protocols have attracted much attention as consequential security issues, and numerous protocols have been studied. In 2017, Bae et al. presented a smartcard-based two-factor authentication protocol for multi-gateway IoT environments. However, we point out that Bae et al.'s protocol is vulnerable to user impersonation attacks, gateway spoofing attacks, and session key disclosure, and cannot provide a mutual authentication. In addition, we propose a three-factor mutual authentication protocol for multi-gateway IoT environments to resolve these security weaknesses. Then, we use Burrows–Abadi–Needham (BAN) logic to prove that the proposed protocol achieves secure mutual authentication, and we use the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool to analyze a formal security verification. In conclusion, our proposed protocol is secure and applicable in multi-gateway IoT environments.

Keywords: internet of things; multi-gateway; mutual authentication; cryptanalysis, BAN logic; AVISPA

---

1. Introduction

Internet of Things (IoT) provides numerous types of services through the internet to exchange data among sensors, embedded systems, and mobile devices. In recent years, IoT environments such as smart buildings, smart factories, smart homes, and smart offices are rapidly becoming a part of our life. A typical IoT architecture consists of heterogeneous micro devices and collects various types of information in real time. However, this is not efficient for practical IoT systems because the communication and computation cost can be increased when the size of IoT networks and the distance between participants are expanded [1,2]. The gateway nodes are deployed to enhance the performance, which provides the ability to communicate with each other efficiently. In a multi-gateway IoT environment, many gateway nodes are deployed and it can process the capability of large-scale IoT networks. IoT environments are also vulnerable to various attacks due to the nature of the open communication channel. Malicious attackers may attempt to insert, delete, and modify the data to obtain users' sensitive information and masquerade as valid users. Much research has been done to resolve security problems in IoT environments. Secure mutual authentication is a primitive and essential method to provide secure communication and numerous secure mutual authentication protocols for IoT have been presented to provide various security features [2–16].

In 2017, Bae et al. [15] proposed a smartcard-based secure authentication protocol in multi-gateway IoT environments to reduce the computational and communication cost. However, we demonstrate that Bae et al.'s protocol is vulnerable to user impersonation, gateway spoofing, and trace and session key disclosure attacks, and does not provide anonymity and a secure mutual authentication. Then, we propose a three-factor authentication protocol that is based on the biometric information of the user, for IoT environments. To analyze the security aspects, we perform an informal security analysis and use Burrows–Abadi–Needham (BAN) logic. Furthermore, we perform a formal security verification using Automated Validation of Internet Security Protocols and Applications (AVISPA) software to check that our protocol can resist man-in-the-middle attacks and replay attacks. We compare the computation cost and security features of our proposed protocol with those of related existing protocols.

The remainder of this paper is as follows. In Sections 2 and 3, we introduce related works and our preliminary details. In Sections 4 and 5, we review Bae et al.'s protocol and cryptanalyze its security flaws. Then, we propose a secure three-factor mutual authentication protocol for multi-gateway IoT environments in Section 6. In Section 7, we prove that our proposed protocol provides a secure mutual authentication using BAN logic. We also perform the AVISPA simulation as a formal security verification and compare the computation cost and security properties with related protocols in Sections 8 and 9. Finally, we conclude with the results of this paper in Section 10.

2. Related Works

Various authentication protocols in single server environments have been proposed [3–5]. In 2010, Wu et al. [3] presented a novel authentication protocol for the telecare medical information system (TMIS). Their protocol provides a guarantee to legitimate users. However, Debiao et al. [6] demonstrated that Wu et al.'s protocol cannot withstand several attacks such as impersonation, replay, or man-in-the-middle attacks. Debiao et al. proposed a more safe and efficient remote authentication protocol for TMIS. In 2013, Chang et al. proposed a secure authentication protocol that provided users privacy. But, in 2103, Das et al. [7] showed that their protocol cannot provide several security features and proper authentication. Furthermore, these authentication protocols are not suitable for distributed systems that consist of multiple servers, such as IoT environments, because the users who want to access the IoT services have to know as many identities and passwords as the number of servers [8,9]. In addition, the physical performance of a single server has limitations [17], and IoT environments are resource-constrained. Therefore, multi-gateway (multi-server) IoT environments are more efficient and useful than the traditional IoT structure [1,2,10,13–16].

In 2014, Turkanovic et al. [5] presented an authentication protocol for IoT environments. However, in 2016, Amin and Biswas [10] pointed out that Turkanovic et al.'s protocol does not withstand several attacks such as offline identity and password guessing, impersonation, and stolen smartcard attacks. They also demonstrated that Turkanovic et al.'s protocol has an inefficient authentication phase. Then, Amin and Biswas proposed an authentication protocol for multi-gateway wireless sensor networks. In 2017, Wu et al. [1] proved that Amin and Biswas's protocol does not resist sensor capture, offline guessing, session key disclosure, impersonation, and desynchronization attacks. They also proved that Amind and Biswas's protocol does not withstand user tracking attacks and does not achieve mutual authentication. Then, Wu et al. proposed a mutual authentication and key agreement protocol for multi-gateway wireless sensor network in IoT. In the same year, Srinivas et al. [13] also proved that Amin and Biswas's protocol has security flaws. Srinivas et al. pointed out that sensor devices have low power, limited memory, and limited battery. Thereafter, Srinivas et al. proposed a more secure and efficient remote user authentication protocol for multi-gateway wireless sensor networks that are suitable for IoT environments.

In 2016, Das et al. [10] presented a three-factor multi-gateway-based user authentication protocol for wireless sensor networks. Das et al. suggested the multi-gateway environment for wireless sensor networks because the generalized wireless sensor networks can bring a lot of overhead to the gateway and have more power consumption than multi-gateway-based wireless sensor networks.

They demonstrated that their protocol can withstand attacks such as sensor capture, privileged-insider, offline password guessing, and impersonation attacks. However, Wu et al. [1] pointed out that Das et al.'s protocol does not resist user tracking attacks and does not have a same session key for all three participants.

In 2018, Wu et al. [14] proposed an authentication protocol for healthcare systems in multi-gateway wireless medical sensor networks. Their protocol prevents malicious attacks such as patient tracking, insider, and offline guessing attacks. Wu et al. demonstrated that multi-gateway environments are suitable for collecting patients' health data through wireless health sensors because the gateway in each area collects the information of patients in the area and then sends it to the doctor. They also demonstrated that their protocol is suitable for transferring data with low time and communication costs.

In 2017, Bae et al. [15] proposed a smartcard-based secure authentication protocol in multi-gateway IoT environments to reduce the computational and communication cost. However, their protocol does not resist impersonation, gateway spoofing, traceability, and session key disclosure attacks and does not guarantee secure mutual authentication and anonymity.

3. Preliminaries

In this section, we introduce a threat model for cryptanalyzing Bae et al.'s protocol, the fuzzy extraction that we use for the cryptographic system in our authentication protocol, and the system model of our protocol in multi-gateway IoT environments. Finally, we present the notations used in this paper.

3.1. Threat Model

We adopt the Dolev–Yao (DY) threat model [18] to analyze Bae et al.'s protocol and our proposed protocol. This model is popularly applied to estimate security. The general assumptions of the DY threat model are as below:

- An attacker can eavesdrop, delete, modify, or insert the transmitted messages via an insecure channel.
- An attacker can steal the smartcard or use a lost smartcard to extract the sensitive information stored in the smartcard [19].
- An attacker can perform various attacks such as trace, impersonation, smartcard lost, man-in-the-middle, replay attacks, and so on.

3.2. Fuzzy Extraction

We briefly show a description of the fuzzy extractor [20] that can extract key information from the given biometric data of users. Biometric information is weak to noises and it is hard to reproduce the actual biometrics from biometric templete in common practice. Moreover, the hash function is sensitized to input, so completely different outputs may come out. Because of these problems, we use the fuzzy extractor method [21,22], which is a type of key generating designed to convert noisy data to public information and a secret random string. The fuzzy extractor restores the original biometric information for noisy biometric data using public help information. The algorithms of the fuzzy extractor are as follows:

- $Generate(BIO_i) =< R_i, P_i >$. This algorithm is for generating key information. It uses biometric data BIO_i as an input and then outputs secret key data R_i, which is a uniformly random string, and a public reproduction P_i as a helper string.
- $Reproduce(BIO_i^{'}, P_i) = R_i$. This algorithm reproduces the secret data R_i. The inputs of this algorithm are a noisy biometric $BIO_i^{'}$ and P_i. The algorithm reproduces the secret biometric key R_i. To recover the same R_i, the metric space distance between BIO_i and $BIO_i^{'}$ should be within a given error tolerance.

3.3. System Model

We introduce a system model of with our proposed protocol for multi-gateway IoT environments. The model consists of three entities: Users, Gateways, and a Control Server. The multi-gateway IoT system model is illustrated in Figure 1.

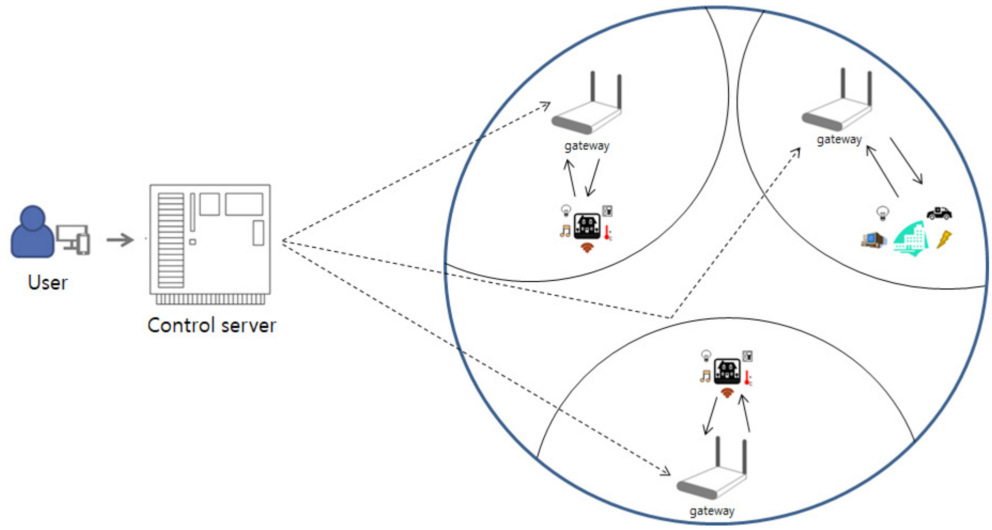

Figure 1. System model of our protocol in multi-gateway IoT environments.

- Users: A user who wants to use the IoT service receives a smartcard from the control server to access the multi-gateway. After registration, login, and authentication, the user has access to use the IoT service. The users' smartcard can be lost or stolen by an attacker.
- Gateways: The gateways consist of IoT environments such as smart homes, smart buildings, smart offices, and gateways. We assume that the gateway and IoT environments are connected in advance by a wireless network through a secure authentication. The performance of the gateways is approximately the performance of the server computer.
- Control Server: The control server is a trusted authentication server with sufficient computation power to compute complicated hash and exclusive functions or store security parameters. The control server stores the identities of the legitimate gateways in advance, and we assume that an attacker can never attack the control server.

3.4. Notations

Table 1 shows the notations used in this paper.

Table 1. Notations.

Notations	Meanings
U_i	i-th user
S_j	j-th server
CS	Control server
ID_i	Identity of U_i
SID_j	Identity of S_j
PW_i	Password of U_i
x	Master secret key chosen by CS
Ts	Timestamp
N_{i1}	Random number generated by U_i's smartcard
N_{i2}	Random number generated by S_j
N_{i3}	Random number generated by CS
SK	Common session key shared among U_i, S_j, and CS
$h(*)$	Collision-resistant one-way hash function

4. Review of Bae et al.'s Protocol

In this section, we overview Bae et al.'s authentication protocol in multi-gateway IoT environments, which consists of three phases: user and server registration phase, user login and authentication phase, and password update phase. In Bae et al.'s protocol, they assumed that the authentication server CS is trusted.

4.1. Registration Phase

If a new user U_i or server S_j requests registration to the authentication server CS, CS issues the smartcard to U_i and sends the necessary value to S_j. This phase and verifier table is shown in Figure 2 and Table 2, respectively, and the details are as follows.

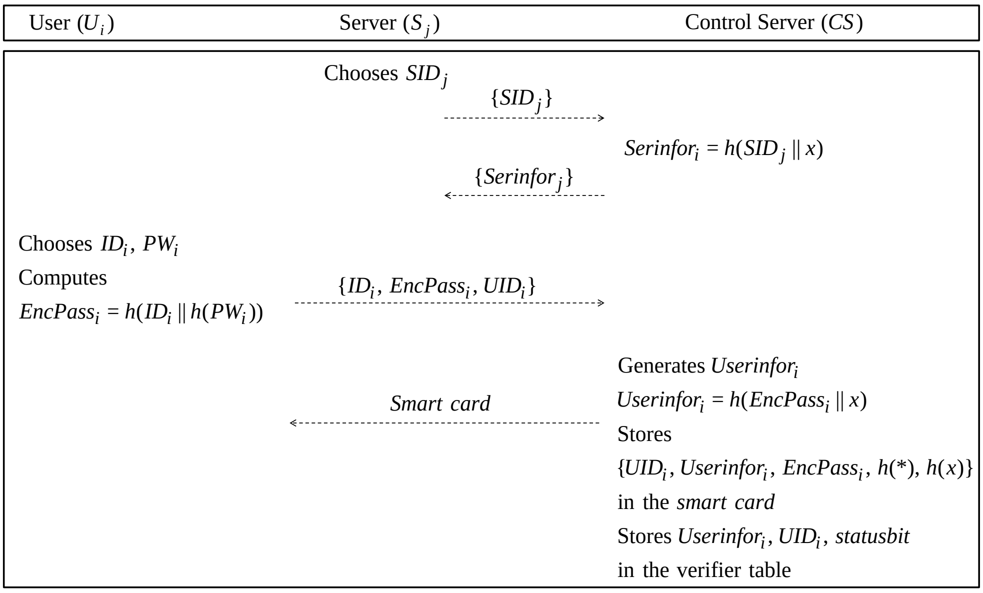

Figure 2. Registration phase of Bae et al.'s protocol.

Step 1: S_j requests registration to the CS. S_j sends its identity SID_j to CS through a secure channel, then CS computes $Serinfor_j$ and sends this to S_j.

Step 2: U_i chooses the ID_i, and PW_i, computes $EncPass_i = h(ID_i||h(PW_i))$ and sends the message $(ID_i, EncPass_i)$ and UID_i, which is an anonymity value of U_i, to CS through a closed channel.

Step 3: CS receives the message from U_i. CS computes the secret information value $Userinfor_i = h(EncsPass_i \parallel x)$, stores $\{UID_i, Userinfor_i, EncPass_i, h(*), h(x)\}$ in the smartcard, and stores $Userinfor_i$, UID_i and *statusbit* in the verifier table. Then, CS issues the smartcard to U_i.

Table 2. The verifier table.

User-Verifier	Anonymity Value	Status-Bit
$Userinfor_1$	U_1	0/1
$Userinfor_2$	U_2	0/1
...	...	...
$Userinfor_i$	U_i	0/1

4.2. Login and Authentication Phase

User U_i must send a login request message to S_j to use the service of server S_j. After receiving a request message, S_j sends a login request message to control server CS. This phase is illustrated in Figure 3 and the following details.

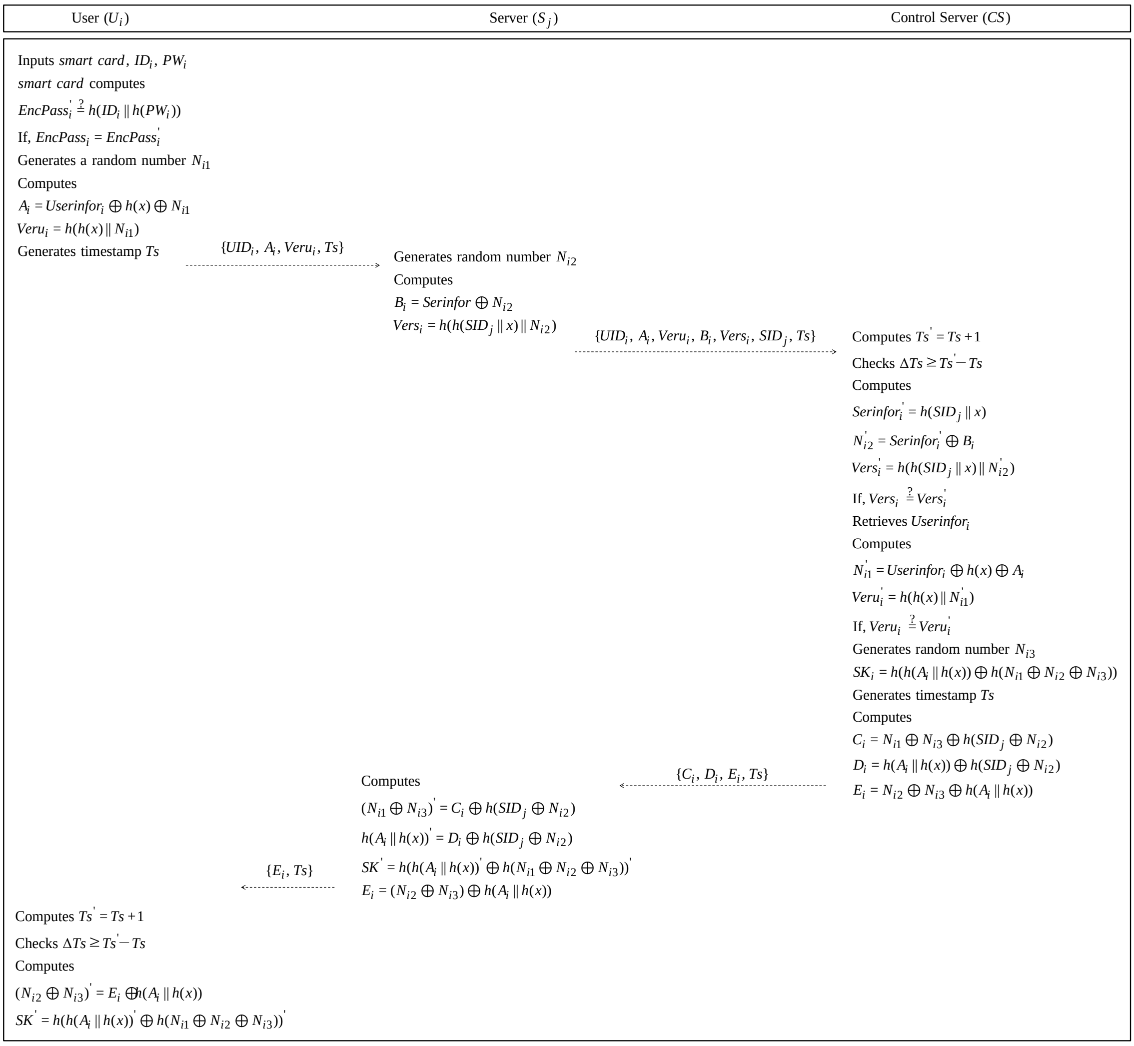

Figure 3. Login and authentication phase of Bae et al.'s protocol.

Step 1: U_i inputs his/her ID_i and PW_i and inputs the smartcard into a smartcard reader. The smartcard computes $EncPass_i' = h(ID_i||h(PW_i))$. Then, the smartcard checks whether $EncPass_i \stackrel{?}{=} EncPass_i'$. If it is equal, U_i generates a random number N_{i1} and computes $A_i = Userinfor_i \oplus h(x) \oplus N_{i1}$, $Veru_i = h(h(x)||N_{i1})$. Then, U_i generates Ts to prevent a replay attack. Finally, U_i sends the login request message $\{UID_i, A_i, Veru_i, Ts\}$ to S_j through a secure channel.

Step 2: If S_j receives the login request message, S_j generates a random number N_{i2} and computes $B_i = Serinfor_j \oplus N_{i2}$, $Vers_i = h(h(SID_j||x)||N_{i2})$. Then, S_j sends the login request message $\{UID_i, A_i, Veru_i, B_i, Vers_i, SID_j, Ts\}$ to CS through an open channel.

Step 3: After CS receives the login request message from S_j, CS computes $Ts' = Ts + 1$ and checks $\Delta Ts \geq Ts' - Ts$ to see whether the login request message is legitimate. If it is valid, CS computes $Serinfor_j' = h(SID_j||x)$, $N_{i2}' = Serinfor_i' \oplus B_i$, $Vers_i' = h(h(SID_j||x)||N_{i2}')$. Then CS compares $Vers_i \stackrel{?}{=} Vers_i'$ to check that the message from S_j is valid. If it is equal, CS retrieves $Userinfor_i$ from the verifier table using UID_i from the login request message. Then, CS computes $N_{i1}' = Userinfor_i \oplus h(x) \oplus A_i$, $Veru_i' = h(h(x)||N_{i1}')$. If $Veru_i \stackrel{?}{=} Veru_i'$ is correct, CS selects a random number N_{i3} and generates a session key $SK_i = h(h(A_i||h(x)) \oplus h(N_{i1} \oplus N_{i2} \oplus N_{i3}))$. CS generates time stamp Ts and computes

$C_i = N_{i1} \oplus N_{i3} \oplus h(SID_j \oplus N_{i2})$, $D_i = h(A_i||h(x)) \oplus h(SID_j \oplus N_{i2})$, $E_i = N_{i2} \oplus N_{i3} \oplus h(A_i||h(x))$. Finally, CS sends an authentication message $\{C_i, D_i, E_i, Ts\}$ to S_j.

Step 4: After S_j receives the message from CS, S_j computes $(N_{i1} \oplus N_{i3})' = C_i \oplus h(SID_j \oplus N_{i2})$, $h(A_i||h(x))' = D_i \oplus h(SID_j \oplus N_{i2})$. S_j generates a session key $SK' = h(h(A_i||h(x))' \oplus h(N_{i1} \oplus N_{i2} \oplus N_{i3}))'$. Then, S_j computes $E_i = (N_{i2} \oplus N_{i3}) \oplus h(A_i||h(x))$ and sends an authentication message $\{E_i, Ts\}$ to U_i.

Step 5: After receiving the message from S_j, U_i computes $Ts' = Ts + 1$ and checks whether $\Delta Ts \geq Ts' - Ts$. If it is correct, U_i computes $(N_{i2} \oplus N_{i3})' = E_i \oplus h(A_i||h(x))$ and generates a session key $SK'' = h(h(A_i||h(x)) \oplus h(N_{i1} \oplus N_{i2} \oplus N_{i3}))'$. Therefore, U_i, S_j, and CS generate the same session key, so they can perform the authentication.

4.3. Password Change Phase

If U_i wants to change his/her password PW_i to a new password PW_i^{new}, the password change phase is performed. This phase is illustrated in Figure 4 and is described as follows.

Step 1: The U_i inserts his/her smartcard into a card reader and inputs ID_i and PW_i. Then, U_i sends the $\{ID_i, PW_i\}$ to the smartcard reader through the closed channel.

Step 2: After receiving the values from U_i, the smartcard computes $EncPass_i = h(ID_i||h(PW_i))$, $Userinfor_i' = h(EncPass_i||x)$. The smartcard verifies whether $Userinfor_i' \stackrel{?}{=} Userinfor_i$. If it is equal, the smartcard requests a new password.

Step 3: U_i inputs a new password PW_i^{new} and generates $EncPass_i^{new} = h(ID_i||h(PW_i^{new}))$. Then, U_i inputs $EncPass_i^{new}$ into the smartcard.

Step 4: The smartcard computes $Userinfor_i^{new} = h(EncPass_i^{new}||x)$ by using $EncPass_i^{new}$. The smartcard updates $Userinfor_i$ to $Userinfor_i^{new}$ and replaces $Userinfor_i$. Finally, the user U_i changes his/her password.

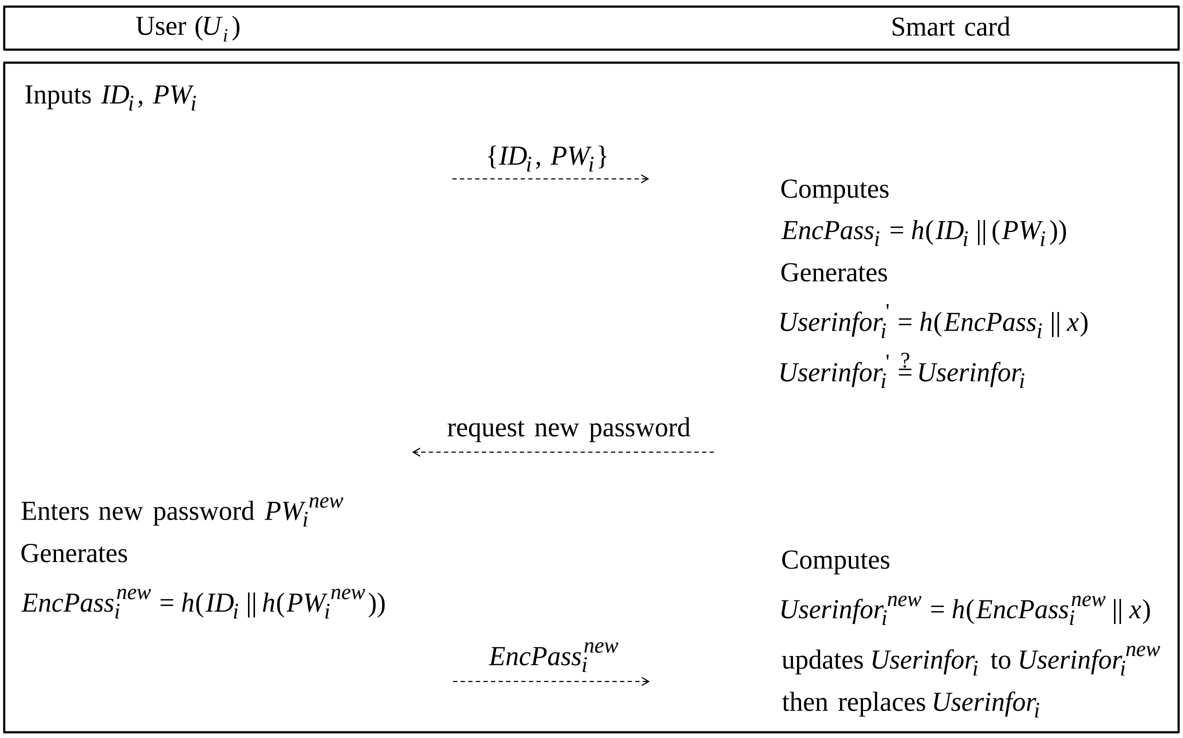

Figure 4. Password change phase of Bae et al.'s protocol.

5. Cryptanalysis of Bae et al.'s Protocol

We analyze the security flaws of Bae et al.'s protocol in this section. Bae et al. asserted that their proposed protocol can prevent various attacks such as user impersonation, server spoofing, and session key disclosure attacks. However, we demonstrate that their protocol does not prevent the following attacks.

5.1. User Impersonation Attack

If an attacker U_a attempts to impersonate an authorized user U_i, U_a must successfully compute a login request message $\{UID_i, A_i, Veru_i, Ts\}$. According to Section 3.1, we can assume that U_a extracts the values $\{UID_i, Userinfor_i, EncPass_i, h(x)\}$ from the smartcard of U_i and obtains the transmitted messages over a public channel. After that, U_a can impersonate the user in the following steps.

Step 1: U_a obtains $\{Userinfor_i, h(x)\}$, $\{A_i, Ts\}$ from the smartcard of U_i and the previous session, respectively.

Step 2: U_a computes $N_{i1} = A_i \oplus Userinfor_i \oplus h(x)$ and obtains a random nonce N_{i1}. Then U_a computes $Veru_i = h(h(x)||N_{i1})$.

Step 3: U_a computes $A_i = Userinfor_a \oplus h(x) \oplus N_{a1}$, $Veru_a = h(h(x)||N_{a1})$. Finally, U_a can generate a login request message $\{UID_i, A_i, Veru_a, Ts\}$ successfully.

5.2. Server Spoofing Attack

To obtain the sensitive information of a user, an attacker attempts to impersonate the server. Bae et al. asserted that their protocol can withstand server spoofing attacks. However, we analyze that their protocol does not resist server spoofing. First, an attacker U_a obtains message $\{E_i, Ts\}$ and extracts the information $h(x)$ from the smartcard of an authorized user. Then, U_a can impersonate the server by generating authentication messages in the following steps.

Step 1: U_a obtains transmitted messages $\{E_i, Ts\}$ in the previous session and extracts $h(x)$ from the smartcard of an authorized user.

Step 2: U_a computes $h(A_i||h(x))$ and obtains $(N_{i2} \oplus N_{i3})$. After that, U_a computes $E_i = (N_{i2} \oplus N_{i3}) \oplus h(A_i||h(x))$.

Step 3: Finally, U_a generates authentication messages $\{E_i, Ts\}$ successfully.

5.3. Session Key Disclosure Attack

Bae et al. demonstrated that their protocol can resist session key disclosure attacks because an attacker cannot compute the values N_{i1}, N_{i2}, and N_{i3}. Furthermore, Bae et al. claimed that the attacker cannot obtain $h(x)$ because the trusted party CS generated $h(X)$. However, we demonstrate that the attacker can compute N_{i1} and $N_{i2} \oplus N_{i3}$ and extract $h(x)$ in Sections 5.1 and 5.2. Thus, the attacker can compute $SK_i = h(h(A_i||h(x)) \oplus h(N_{i1} \oplus N_{i2} \oplus N_{i3}))$. Therefore, Bae et al.'s protocol is vulnerable to session key disclosure attacks.

5.4. Mutual Authentication

In Bae et al.'s protocol, CS computes $Vers_i^{'}$ and $Veru_i^{'}$ to authenticate legitimate U_i and S_j. However, CS cannot generate authentication messages for U_i and S_j. Thus, U_i and S_j receive the message from CS, but they cannot trust the messages because they cannot check whether the attacker sends the message. Therefore, Bae et al.'s protocol does not achieve mutual authentication.

6. A Secure Three-Factor Mutual Authentication Protocol

In this section, we propose a three-factor mutual authentication protocol for multi-gateway IoT environments according to Section 3.3. The proposed protocol consists of three phases: users and gateways registration, login and authentication, and password update.

6.1. Registration Phase

First, a gateway G_j must register with control server CS to provide their services to users. Then, a new user U_i first accesses the control server, and he/she must register with CS. The detailed steps are illustrated in Figure 5 and described as follows.

Step 1: G_j requests registration to the CS. G_j selects GID_j and sends the value to CS through a secure channel, then CS computes $PID_j = h(GID_j||h(x||y))$ and sends PID_j to G_j via a secure channel. G_j stores PID_j in itself.

Step 2: U_i chooses the his/her identification ID_i and password PW_i and imprints biometrics BIO_i. Then U_i generates a random number a_i, computes $< R_i, P_i >= Gen(BIO_i)$, $HIDi = h(ID_i||a_i))$, which is an anonymity value of U_i, and $HPW_i = h(ID_i||PW_i||a_i)$, and sends the message $\{HID_i, HPW_i, a_i\}$ to CS through closed channel.

Step 3: After CS receives the message from U_i, CS computes the secret information value $UI_i = h(HID_i||a_i||x)$, $A_i = UI_i \oplus h(HPW_i)$, $B_i = h(UI_i||A_i)$, and $X_i = h(UI_i||x)$. Then, CS stores $\{A_i, B_i, X_i, h(*)\}$ in the smartcard, and stores UI_i with HID_i in the database. Then CS issues the smartcard to U_i.

Step 4: After receiving the smartcard from CS, U_i computes $L_i = h(R_i||PW_i) \oplus a_i$. Then U_i inputs L_i and P_i in the smartcard.

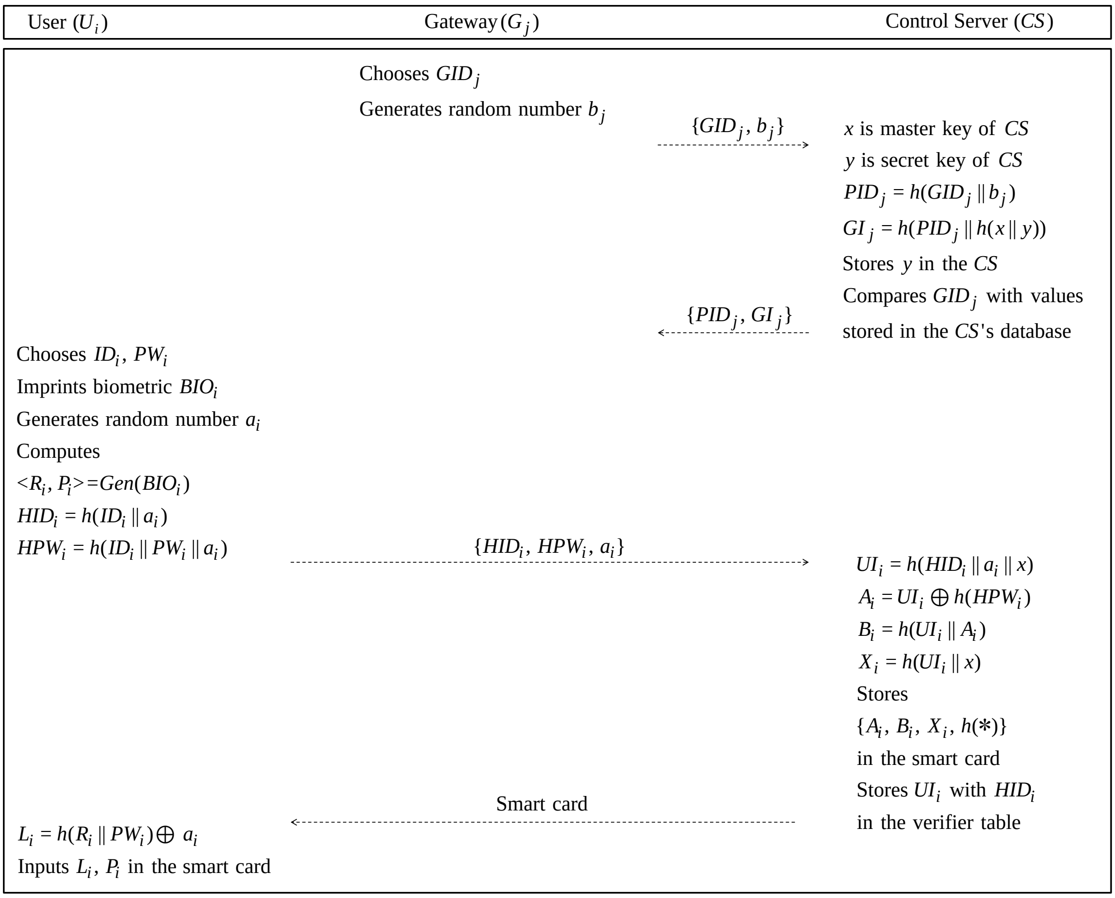

Figure 5. Registration phase of our proposed protocol.

6.2. Login and Authentication Phase

If a user U_i wants to use the service of gateway G_j, U_i must send a login request message to G_j. Then, G_j sends a login request message to control server CS. The detailed steps are illustrated in Figure 6 and described as follows.

User (U_i)	Gateway (G_j)	Control Server (CS)
Inputs *smart card*, ID_i, PW_i and biometric BIO_i *Smart card* computes $R_i = Rep(BIO_i, P_i)$ $a_i = L_i \oplus h(R_1 \parallel PW_i)$ $HID_i = h(ID_i \parallel a_i)$ $HPW_i = h(ID_i \parallel PW_i \parallel a_i)$ $UI_i = A_i \oplus h(HPW_i)$ $B_i^* = h(UI_i \parallel A_i)$ If, $B_i^* \stackrel{?}{=} B_i$ Generates a random number N_i Computes $C_i = UI_i \oplus N_i$ $VU_i = h(X_i \parallel N_i \parallel GID_j)$		
$\xrightarrow{\{HID_i, C_i, VU_i\}}$	Generates random number N_j Computes $D_i = GI_j \oplus N_j$ $VS_j = h(GID_j \parallel GI_j \parallel N_j)$	
	$\xrightarrow{\{HID_i, C_i, VU_i, PID_j, D_i, VS_j\}}$	Computes $GI_j = h(PID_j \parallel h(x \parallel y))$ $N_j = D_i \oplus GI_j$ $VS_j^* = h(GID_j \parallel GI_j \parallel N_j)$ If, $VS_j^* \stackrel{?}{=} VS_j$ Retrieves UI_i Computes $X_i = h(UI_i \parallel x)$ $N_i = C_i \oplus UI_i$ $VU_i^* = h(X_i \parallel N_i \parallel GID_j)$ If, $VU_i^* \stackrel{?}{=} VU_i$ Generates a random number N_c $E_i = GI_j \oplus N_c$ $F_i = GI_j \oplus N_j$ $M_{cg} = h(E_i \parallel GI_j \parallel N_c)$ $SK = h(N_i \oplus h(N_j \parallel N_c))$ $M_{cu} = h(X_i \parallel UI_i \parallel N_i)$ Updates HID_i, UI_i to HID_i^{new}, UI_i^{new} then replaces HID_i, UI_i $HID_i^{new} = h(HID_i \parallel N_i \parallel h(N_j \parallel N_c))$ $UI_i^{new} = h(HID_i^{new} \parallel N_i \parallel UI_i)$
	$N_c = E_i \oplus GI_j$ $M_{cg}^* = h(E_i \parallel GI_j \parallel N_c)$ *If*, $M_{cg}^* \stackrel{?}{=} M_{cg}$ *Computes* $N_i = F_i \oplus GI_j$ $SK = h(N_j \oplus h(N_j \parallel N_c))$ $G_i = h(GID_j \parallel N_i)$ $H_i = G_i \oplus h(N_j \parallel N_c)$	$\xleftarrow{\{M_{cg}, M_{cu}, E_i, F_i\}}$
$M_{cu}^* = h(X_i \parallel UI_i \parallel N_i)$ *If*, $M_{cu}^* \stackrel{?}{=} M_{cu}$ *Computes* $G_i^* = h(GID_j \parallel N_i)$ $h(N_j \parallel N_c) = H_i \oplus G_i^*$ $SK = h(N_i \oplus h(N_j \parallel N_c))$ Updates HID_i, UI_i to HID_i^{new}, UI_i^{new} then replaces HID_i, UI_i $HID_i^{new} = h(HID_i \parallel N_i \parallel h(N_j \parallel N_c))$ $UI_i^{new} = h(HID_i^{new} \parallel N_i \parallel UI_i)$ Updates $A_i^{new} = UI_i^{new} \oplus h(HPW)$ $B_i^{new} = h(UI_i^{new} \parallel A_i^{new})$ $X_i^{new} = h(UI_i^{new} \parallel UI_i)$	$\xleftarrow{\{H_i, M_{cu}\}}$	

Figure 6. Login and authentication phase of our proposed protocol.

Step 1: U_i inserts the smartcard, his/her ID_i and PW_i, and biometric BIO_i. The smartcard computes $R_i = Rep(BIO_i, Pi)$, $a_i = L_i \oplus h(R_i||PW_i)$, $HID_i = h(ID_i||a_i)$, $HPW_i = h(ID_i||PW_i||a_i)$, $UI_i = A_i \oplus h(HPW_i)$, $B_i^* = h(UI_i||A_i)$. Then, the smartcard checks whether $B_i^* \stackrel{?}{=} B_i$ to check whether the user is legitimate. If it is valid, U_i generates a random number N_i and computes $C_i = UI_i \oplus N_i$, $VU_i = h(X_i||N_i||GID_j)$. Finally, U_i sends the login request message $\{HID_i, C_i, VU_i\}$ to G_j through a public channel.

Step 2: After receiving a login request message, G_j generates a random number N_j and computes $D_i = GI_j \oplus N_j$, $VS_j = h(GID_j||GI_j||N_j)$. Then, G_j sends the login request message $\{HID_i, C_i, PID_j, D_i, VS_j\}$ to CS via an open channel.

Step 3: After CS receives the login request message from G_j, CS computes $GI_j = h(PID_j||h(x||y))$, $N_j = D_i \oplus GI_j$ and compares $VS_j^* \stackrel{?}{=} VS_j$ to see whether G_j's login request message is legitimate. If it is equal, CS retrieves UI_i from the verifier table using HID_i of the login request message. Then, CS computes $X_i = h(UI_i||x)$, $N_i = C_i \oplus UI_i$, $VU_i^* = h(X_i||N_i||GID_j)$. Then CS compares $VU_i^* \stackrel{?}{=} VU_i$ to check that the message from U_i is valid. If it is valid, CS generates a random number N_c and computes $E_i = GI_j \oplus N_c$, $F_i = GI_j \oplus N_i$. CS computes $M_{cg} = h(E_i||GI_j||N_c)$ to mutually authenticate with $G_j$and $M_{cu} = h(X_i||UI_i||N_i)$ to authenticate with U_i and generates a session key $SK = h(N_i \oplus h(N_j||N_c))$. CS updates HID_i to $HID_i^{new} = h(HID_i||N_i||h(N_j||N_c))$ and UI_i to $UI_i^{new} = h(HID_i^{new}||N_i||UI_i)$, then replaces HID_i and UI_i. Finally, CS sends the authentication message $\{M_{cg}, M_{cu}, E_i, F_i\}$ to G_j.

Step 4: After G_j receives the authentication message from CS, G_j computes $N_c = E_i \oplus GI_j$, $M_{cg}^* = h(E_i||GI_j||N_c)$.Then, G_j compares $M_{cg}^* \stackrel{?}{=} M_{cg}$ to verify whether the message from CS is legitimate. If it is valid, G_j computes $N_i = F_i \oplus GI_j$ and generates a session key $SK = h(N_i \oplus h(N_j||N_c))$. Then, G_j computes $G_i = h(GID_j||N_i)$, $H_i = G_i \oplus h(N_j||N_c)$ and sends the authentication message $\{H_i, M_{cu}\}$ to U_i.

Step 5: After receiving the message from G_j, U_i computes $M_{cu}^* = h(X_i||UI_i||N_i)$ and verifies whether $M_{cu}^* \stackrel{?}{=} M_{cu}$. If it is valid, U_i computes $G_i^* = h(GID_j||N_i)$, $h(N_j||N_c) = H_i \oplus G_i^*$ and generates a session key $SK = h(N_i \oplus h(N_j||N_c))$. Therefore, U_i, S_j, and CS generate the same session key, so they can perform the authentication. U_i updates HID_i to $HID_i^{new} = h(HID_i||N_i||h(N_j||N_c))$ and UI_i to $UI_i^{new} = h(HID_i^{new}||N_i||UI_i)$, then replaces HID_i and UI_i. The smartcard updates $A_i^{new} = UI_i^{new} \oplus h(HPW)$, $B_i^{new} = h(UI_i^{new}||A_i^{new})$, and $X_i^{new} = h(UI_i^{new}||UI_i)$.

6.3. Password Change Phase

If U_i wants to change his/her password, U_i performs the password change phase without the help of G_j. The detailed steps of the password change phase are shown in Figure 7 and described as follows.

Step 1: A legitimate user U_i inserts the smartcard, his/her ID_i and PW_i, and biometric BIO_i.

Step 2: The smartcard computes $< R_i, P_i >= Gen(BIO_i)$, $a_i = L_i \oplus h(R_i||PW_i)$, $HPW_i = h(ID_i||PW_i||a_i)$, and $B_i^* = h(UI_i||A_i)$. After that, the smartcard compares the B_i^* with B_i stored value. If it is equal, the smartcard requests a new password to U_i.

Step 3: When U_i receives the request message from smartcard, U_i inputs a new password PW_i^{new}.

Step 4: After receiving the new password from U_i, the smartcard computes $L_i^{new} = a_i \oplus h(R_i||PW_i^{new})$, $HPW_i^{new} = h(ID_i||PW_i^{new}||a_i)$, $A_i^{new} = UI_i \oplus h(HPW_i^{new})$, and $B_i^{new} = h(UI_i||A_i^{new})$. Consequently, the smartcard updates the old information $\{A_i, B_i, L_i\}$ to new information $\{A_i^{new}, B_i^{new}, L_i^{new}\}$.

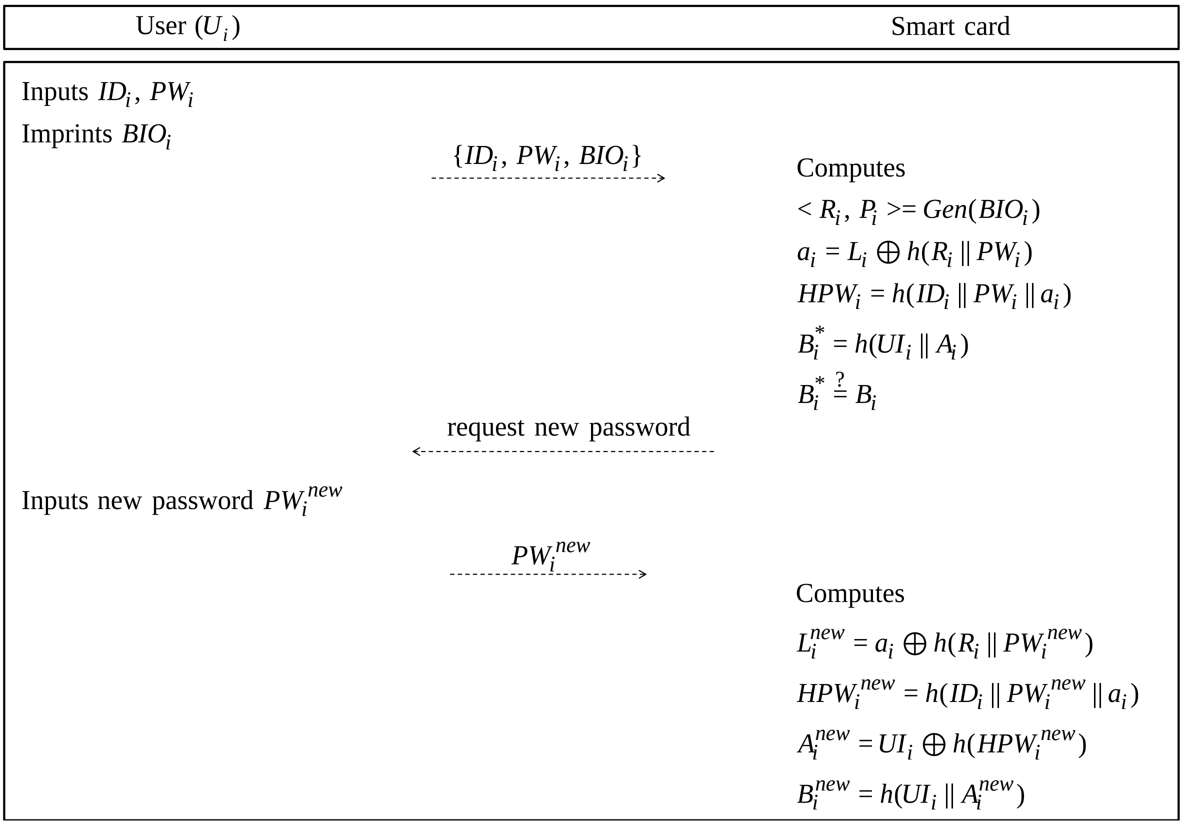

Figure 7. Password change phase of our proposed protocol.

7. Security Analysis

We show that our proposed protocol can prevent various attacks by performing an informal analysis, as mentioned in Section 3.1. We analyze our protocol using Burrows–Abadi–Needham (BAN) logic to prove that our protocol can achieve secure mutual authentication.

7.1. Informal Security

To prove that our proposed protocol can prevent various attacks such as trace, smartcard lost, impersonation, off-line guessing, and session key disclosure attacks, we perform an informal security analysis. Additionally, we show that proposed protocol provides anonymity and a secure mutual authentication.

7.1.1. User Impersonation Attack

If a malicious attacker U_a attempts to masquerade as a user U_i, U_a can generate a login request message $\{HID_i, C_i, VU_i\}$ and message $\{H_i, M_{cu}\}$. However, U_a cannot compute HID_i because U_a cannot extract a random number a_i from HID_i. U_a cannot retrieve a random number N_i because the attacker cannot know secret parameter UI_i. Thus, U_a cannot compute C_i, VU_i because U_a cannot extract a random number N_i. Therefore, our protocol resists user impersonation attack.

7.1.2. Server Spoofing Attack

To impersonate the server, an attacker U_a can generate an authentication message $\{H_i, M_{cu}\}$. However, U_a cannot compute these because U_a cannot know the random nonces N_i, N_j, N_c. Furthermore, if U_a attempts to impersonate the gateway by using public parameter GID_j, the control server compares it with the stored identities of the legitimate gateways in advance. Thus, our proposed protocol is secure against server spoofing attacks because U_a cannot generate valid messages.

7.1.3. Smartcard Stolen Attack

We assume that an attacker U_a can extract the values of the smartcard $\{A_i, B_i, X_i, L_i, h(*)\}$ according to Section 3.1. However, U_a cannot obtain sensitive or useful information without the identity, password, and biometrics of the legitimate user because the values stored in the smartcard are safeguarded with a one-way hash function or an XOR operation of $ID_i, PW_i, HPW_i = h(ID_i||PW_i||a_i)$. Therefore, our protocol can prevent smartcard stolen attacks.

7.1.4. Trace Attack and Anonymity

In our protocol, an attacker U_a cannot know the identity of the users and gateways. The user U_i does not send a real identity ID_i via the public channels. The user generates and sends a pseudonym identity $HID_i = h(ID_i||a_i)$. Because HID_i is a transmitted message via a public channel, U_a can obtain this value. Therefore, U_i updates it as $HID_i^{new} = h(HID_i||N_i||h(N_j||N_c))$ for every session to prevent the attack of U_a. The gateway uses PID_j, which is generated in the registration phase, instead of GID_j, so our protocol provides anonymity of users and gateways. In addition, the proposed protocol resists trace attacks because all messages are dynamic for every session.

7.1.5. Man-in-the-Middle Attack and Replay Attack

We assume that attacker U_a knows the information transmitted via an insecure channel and information from the smartcard of U_i to set up a secure channel with G_j. However, U_a cannot generate a valid login request message, as mentioned. Furthermore, U_a cannot impersonate user U_i by resending the messages because the messages are refreshed with random numbers N_i, N_j, and N_c. Therefore, our proposed protocol prevents man-in-the-middle attacks and replay attacks.

7.1.6. Off-Line Password Guessing Attack

An attacker U_a attempts to guess the password PW_i of legitimate user U_i. If U_a can guess the password, U_a can compute a series of equations and compute several equations and the valid value with the guessed passwords. However, U_a must know the unique biometrics of the user to compute equations. Therefore, it is impossible to guess the user's password in our protocol.

7.1.7. Desynchronization Attack

For a desynchronization attack, an adversary disturbs the communication of the login and authentication request message. However, CS uses HID_i to retrieve UI_i after checking message from G_j, and HID_i updates HID_i^{new} after authentication of the request message. Furthermore, an attacker disturbs the response communication to desynchronize HID_i^{new}. Even if the user cannot receive the response message, the user can generate and update HID_i^{new}. Thus, our proposed protocol can resist desynchronization attacks.

7.1.8. Mutual Authentication

When control server CS receives the login request message from gateway G_j, CS computes VS_j^* and VU_i^* to authenticate user U_i and G_j. If VS_j and VS_j^* are equal, CS authenticates G_j. Furthermore, CS retrieves U_i from a database to an available VS_j. After that, CS compares VU_i and VU_i^*. If they are equal, CS authenticates U_i. Then, CS computes and sends the login response messages M_{cg} and M_{cu} to authenticate. After receiving M_{cg} from CS, G_j computes M_{cg}^* and compares M_{cg}^* and M_{cg}. If they are equal, G_j authenticates CS. Finally, U_i computes M_{cu}^* and checks whether $M_{cu}^* \stackrel{?}{=} M_{cu}$. If it is valid, U_i authenticates CS. Therefore, U_i, G_j, and CS successfully mutually authenticate. An attacker cannot validate the message, as mentioned in Sections 7.1.1 and 7.1.2. Moreover, the login request and response messages are refreshed for every session according to Sections 7.1.4 and 7.1.5. Therefore, our proposed protocol provides secure mutual authentication.

7.2. Ban Logic

We perform a formal verification to check that our proposed protocol achieves a secure mutual authentication using BAN logic. Table 3 presents the notation of BAN logic. We show the logical rules of BAN logic in Section 7.2.1. In the following sections, we show the goals, idealized forms, and assumptions of our proposed protocol. In Section 7.2.5, we show that our proposed protocol can provide mutual authentication among U_i, G_j, and CS. More details of BAN logic can be found in [23,24].

Table 3. Notations of Burrows–Abadi–Needham (BAN) logic.

Notations	Meaning
$P\| \equiv X$	P believes the statement X
$\#X$	The statement X is fresh
$P \triangleleft X$	P sees the statement X
$P\| X$	P once said X
$P \Rightarrow X$	P controls the statement X
$< X >_Y$	Formula X is combined with formula Y
$\{X\}_K$	Formula X is encrypted by the key K
$P \overset{K}{\leftrightarrow} Q$	P and Q communicate using K as the shared key
SK	Session key used in the current authentication session

7.2.1. Rules of Ban Logic

We introduce rules of BAN logic as follows:

1. Message meaning rule:

$$\frac{P \Big| \equiv P \overset{K}{\leftrightarrow} Q, \quad P \triangleleft \{X\}_K}{P \ |\equiv Q \ | \sim X}$$

2. Nonce verification rule:

$$\frac{P \ |\equiv \#(X), \quad P \ | \equiv Q \Big| \sim X}{P \ |\equiv Q \ | \equiv X}$$

3. Jurisdiction rule:

$$\frac{P \ |\equiv Q \ | \Longrightarrow X, \quad P \ |\equiv Q \ | \equiv X}{P \Big| \equiv X}$$

4. Freshness rule:

$$\frac{P \Big| \equiv \#(X)}{P \Big| \equiv \#(X, Y)}$$

5. Belief rule:

$$\frac{P \Big| \equiv (X, Y)}{P \Big| \equiv X}$$

7.2.2. Goals

We present the following goals to prove that our protocol achieves secure mutual authentication:

Goal 1: $G_j | \equiv CS | \equiv (N_c, N_i)$,
Goal 2: $G_j | \equiv (N_c, N_i)$,
Goal 3: $CS | \equiv G_j | \equiv (N_i, N_j)$,
Goal 4: $CS | \equiv (N_i, N_j)$,

Goal 5: $U_i| \equiv G_j| \equiv (N_c, N_i)$,
Goal 6: $U_i| \equiv (N_j, N_c)$

7.2.3. Idealized Forms

$Msg_1: \ U_i \rightarrow G_j : (HID_i, N_i, x, GID_j)_{UI_i}$

$Msg_2: \ G_j \rightarrow CS : (HID_i, N_i, x, GID_j, N_j)_{GI_j}$

$Msg_3: \ CS \rightarrow G_j : (N_c, N_i, UI_i, x)_{GI_j}$

$Msg_4: \ G_j \rightarrow U_i : (N_c, N_j, UI_i, GID_j, x)_{N_i}$

7.2.4. Assumptions

To achieve the BAN logic proof, we make the following assumptions about the initial state of our proposed protocol:

$A_1: \ G_j| \equiv (U_i \xleftrightarrow{UI_i} G_j)$

$A_2: \ G_j| \equiv \#(N_i)$

$A_3: \ CS| \equiv (G_j \xleftrightarrow{GI_j} CS)$

$A_4: \ CS| \equiv \#(N_j, N_i)$

$A_5: \ G_j| \equiv (G_j \xleftrightarrow{GI_j} CS)$

$A_6: \ U_i| \equiv (U_i \xleftrightarrow{N_i} G_j)$

$A_7: \ U_i| \equiv \#(N_j)$

$A_8: \ CS| \equiv G_j \Rightarrow (CS \xleftrightarrow{GI_j} G_j)$

7.2.5. Proof Using Ban Logic

The following steps are the main proofs using BAN rules and assumptions:

Step 1: According to Msg_1, we can get

$$S_1 : G_j \lhd (HID_i, N_i, x, GID_j)_{UI_i}.$$

Step 2: From A_1 and S_1, we apply the message meaning rule to obtain

$$S_2 : G_j| \equiv U_i \ (HID_i, N_i, x, GID_j)_{UI_i}.$$

Step 3: From A_2 and S_2, we apply the freshness rule to obtain

$$S_3 : G_j| \equiv \#(HID_i, N_i, x, GID_j)_{UI_i}.$$

Step 4: From S_2 and S_3, we apply the nonce verification rule to obtain

$$S_4 : G_j| \equiv U_i \equiv (HID_i, N_i, x, GID_j)_{UI_i}.$$

Step 5: From S_4, we apply the belief rule to obtain

$$S_5 : G_j| \equiv U_i| \equiv (N_i)_{UI_i}.$$

Step 6: According to Msg_2, we can get

$$S_6 : CS \lhd (HID_i, N_i, x, GID_j, N_j)_{GI_j}.$$

Step 7: From A_3 and S_6, we apply the message meaning rule to obtain

$$S_7 : CS| \equiv G_j\ (HID_i, N_i, x, GID_j, N_j)_{GI_j}.$$

Step 8: From A_4 and S_7, we apply the freshness rule to obtain

$$S_8 : CS| \equiv \#(HID_i, N_i, x, GID_j, N_j)_{GI_j}.$$

Step 9: From S_7 and S_8, we apply the nonce verification rule to obtain

$$S_9 : CS| \equiv G_j| \equiv (HID_i, N_i, x, GID_j, N_j)_{GI_j}.$$

Step 10: From S_9, we apply the belief rule to obtain

$$S_{10} : CS| \equiv G_j| \equiv (N_i, N_j)_{GI_j}.\ \textbf{(Goal 3)}$$

Step 11: According to Msg_2, we can get

$$S_{11} : G_j \lhd (N_c, N_i, UI_i, x)_{GI_j}.$$

Step 12: From A_5 and S_{11}, we apply the message meaning rule to obtain

$$S_{12} : G_j| \equiv CS\ (N_c, N_i, UI_i, x)_{GI_j}.$$

Step 13: From A_6 and S_{12}, we apply the freshness rule to obtain

$$S_{13} : G_j| \equiv \#(N_c, N_i, UI_i, x)_{GI_j}.$$

Step 14: From S_{12} and S_{13}, we apply the nonce verification rule to obtain

$$S_{14} : G_j| \equiv CS| \equiv (N_c, N_i, UI_i, x)_{GI_j}.$$

Step 15: From S_{14}, we apply the belief rule to obtain

$$S_{15} : G_j| \equiv CS| \equiv (N_c, N_i)_{GI_j}.\ \ \textbf{(Goal 1)}$$

Step 16: According to Msg_4, we can obtain

$$S_{16} : U_i \lhd (N_c, N_j, UI_i, GID_j, x)_{N_i}.$$

Step 17: From A_6 and S_{16}, we apply the message meaning rule to obtain

$$S_{17} : U_i| \equiv G_j\ (N_c, N_j, UI_i, GID_j, x)_{N_i}.$$

Step 18: From A_7 and S_{17}, we apply the freshness rule to obtain

$$S_{18} : U_i| \equiv \#G_j\ (N_c, N_j, UI_i, GID_j, x)_{N_i}.$$

Step 19: From S_{17} and S_{18}, we apply the nonce verification rule to obtain

$$S_{19}: U_i| \equiv G_j| \equiv (N_c, N_j, UI_i, GID_j, x)_{N_i}.$$

Step 20: From S_{19}, we apply the belief rule to obtain

$$S_{20}: U_i| \equiv G_j| \equiv (N_c, N_j)_{N_i}. \quad \textbf{(Goal 5)}$$

Step 21: From S_{10} and A_8, we apply the jurisdiction rule to obtain

$$S_{21}: CS| \equiv (N_i, N_j). \quad \textbf{(Goal 4)}$$

Step 22: From S_{15} and A_9, we apply the jurisdiction rule to obtain

$$S_{22}: G_j| \equiv (N_c, N_i). \quad \textbf{(Goal 2)}$$

Step 23: From S_{20} and A_{10}, we apply the jurisdiction rule to obtain

$$S_{23}: U_i| \equiv (N_c, N_i). \quad \textbf{(Goal 6)}$$

We show that the proposed protocol can provide secure mutual authentication between U_i, G_j, and CS based on goals 1–6.

8. Formal Verification Using Avispa

We present a formal verification of our proposed protocol using the AVISPA tool based on the High-Level Protocol Specification Language (HLPSL) code [25]. AVISPA is one of the widely used verification tools to check that protocols are secure against man-in-the-middle attacks and replay attacks. Numerous studies have been simulated using the AVISPA tool [26–28]. We will shortly describe AVISPA and show the HLPSL specifications of our proposed protocol. Then, we will assert that the proposed protocol can resist replay and man-in-the-middle attacks through the results of the AVISPA simulation.

8.1. Description of Avispa

AVISPA performs security verification through four back-ends consisting of Constraint-Logic-based Attack Searcher (CL-AtSe) [29], On-the-Fly Model-Checker (OFMC) [30], Tree Automate-Based Protocol Analyzer (TA4SP), and SAT-Based Model-Checker (SATMC). HLPSL specification is translated into intermediate format (IF) by an hlpsl2if translator. IF is converted to the output format (OF), which is produced using the four back-ends as mentioned above. But usually, CL-Atse and OFMC are used for verification. AVISPA has several functions that are mentioned below for analyzing protocols. More details on AVISPA can be found in [31,32].

- $secret(A, id, B)$: id denotes an information A that is only known to B.
- $witness(A, B, id, E)$: id denotes a weakness authentication factor E that is used by A to authenticate B.
- $request(A, B, id, E)$: id denotes a strong authentication factor. B requests A for E to authenticate.

8.2. Hlpsl Specifications of Our Protocol

Our protocol has three basic *roles* which are denoted by entities that have been specified according to HLPSL: UA denotes a user, GA denotes a gateway, and CS denotes a control server. The role of *session* and *environments* are shown in Figure 8. In the *session*, we describe participants. In *environments*, intruder knowledge is defined, and four secrecy goals and four authentication goals are described. The HLPSL specifications of role UA are shown in Figure 9, and the details are as follows.

```
role session(UA, GA, CS : agent, SKuacs, SKgacs :
symmetric_key, H: hash_func)

def=
local SN1, SN2, SN3, RV1, RV2, RV3: channel(dy)
composition
user(UA, GA, CS, SKuacs, H, SN1, RV1)
/\ gateway(UA, GA, CS, SKgacs, H, SN2, RV2)
/\ controlserver(UA, GA, CS,SKuacs, SKgacs, H, SN3,
RV3)
end role
```

```
role environment()
def=
const ua, ga, cs : agent,
skuacs, skgacs: symmetric_key,
h: hash_func,
gidj,hidi,idi: text,
ua_cs_ni, cs_ua_mcu, ga_cs_nj, cs_ga_mcg:
protocol_id,
sp1,sp2,sp3,sp4: protocol_id

intruder_knowledge = {ua,ga,cs,gidj,hidi,idi,h}
composition
session(ua,ga,cs, skuacs, skgacs,h)/\session(i,ga,cs,
skuacs,skgacs, h)
/\session(ua,i,cs, skuacs,skgacs,h)
/\session(ua,ga,i, skuacs,skgacs,h)

end role

goal
secrecy_of sp1, sp2, sp3, sp4
authentication_on ua_cs_ni
authentication_on cs_ua_mcu
authentication_on ga_cs_nj
authentication_on cs_ga_mcg
end goal

environment()
```

Figure 8. Specification of session and environments.

At transition 1, UA starts the registration phase with a start message in state value 0 and then updates the state from 0 to 1. UA sends the registration message $\{HID_i, HPW_i, a\}$ to CS through a closed channel. At transition 2, UA receives the smartcard from CS, then it updates the state from 1 to 2. In state value 2, UA generates the random number N_i, sends the login request message $\{HID_i, C_i, VU_i\}$ to GA via an insecure channel, and declares $witness(UA, CS, us\_cs\_ni, N_i)$, which means that N_i denotes a weakness authentication factor. At transition 3, UA receives the login response message from GA. After that, UA changes the state value from 2 to 3, generates the session key, and declares $request(UA, CS, cs\_ua\_mcu, N_c)$. The specifications of role GA and CS are similar and shown in Figures 10 and 11.

```
role user(UA, GA, CS : agent, SKuacs : symmetric_key, H: hash_func, SND, RCV :
channel(dy))

played_by UA
def=
local State: nat,
   IDi,PWi,BIOi,Ri,Pi,HIDi, HPWi,GIDj,Li, Ai, UIi, Aii, Bi, Ni,Nj,Nc, Ci, Xi, X, Y,
Bj,VUi: text,
   Di,Ei,Fi,Gi,Hi, VSj, Mcg,Mcu, GIj, PIDj : text,
   HIDinew, UIinew, Aiinew, Binew, Xinew : text,
   SKi, SKj, SKc: text
const sp1, sp2, sp3, sp4, ua_cs_ni, cs_ua_mcu, ga_cs_nj, cs_ga_mcg: protocol_id
init State := 0
transition

%%%%%%%%%%%Registration phase
1. State = 0 /\ RCV(start) =|>
State' := 1 /\ Ai' := new() /\ Ri' := new() /\ Pi' := new()
      /\ HIDi' := H(IDi.Ai')
      /\ HPWi' := H(IDi.PWi.Ai')
      /\ SND({HIDi'.HPWi'.Ai'}_SKuacs)
        /\ secret({IDi, PWi, Ri', Pi'}, sp1, {UA})

%%%%%%%%%%%Recieve smartcard
2. State = 1 /\ RCV
({xor(H(H(IDi.PWi.Ai').X),H(H(IDi.PWi.Ai'))).H(H(H(IDi.PWi.Ai').X).xor(H(H(IDi
.PWi.Ai').X),H(H(IDi.PWi.Ai')))).H(H(H(IDi.PWi.Ai').X).X)}_SKuacs)=|>
 State' := 2  /\ Ri' := new() /\ Li':=xor(H(Ri'.PWi),Ai')
%%%%%%%%%%%Login & Authentication phase
       /\ Ni' := new()
       /\ Ci' := xor(H(H(IDi.PWi.Ai').X),Ni')
       /\ VUi' := H(H(H(H(IDi.PWi.Ai').X).X).Ni'.GIDj)
       /\ SND(H(IDi.Ai').Ci'.VUi')
       /\ witness(UA,CS,ua_cs_ni,Ni')
3. State = 2
/\ RCV(xor(H(GIDj.Ni'),H(Nj'.Nc')).H(H(H(H(IDi.PWi.Ai').X).X).H(H(IDi.PWi.Ai').
X).Ni')) =|>
State' := 3 /\ SKi' := H(xor(Ni',H(Nj'.Nc')))
      /\ HIDinew' := H(H(IDi.Ai').Ni'.H(Nj'.Nc'))
      /\ UIinew' := H(H(H(IDi.Ai').Ni'.H(Nj'.Nc')).Ni'.H(H(IDi.Ai').Ai'.X))
      /\ Aiinew' := xor(UIinew', H(H(IDi.PWi.Ai')))
      /\ Binew' := H(UIinew'.Aiinew')
      /\ Xinew' := H(UIinew'.H(H(IDi.Ai').Ai'.X))
      /\ request(UA,CS,cs_ua_mcu,Nc')
end role
```

Figure 9. Specification of user.

8.3. Results of Avispa Simulation

The results of AVISPA simulation through OFMC and CL-AtSe verification are shown in Figure 12. The OFMC and CL-AtSe back-ends check whether our proposed protocol can resist replay attacks and man-in-the-middle attacks. The OFMC verification shows that search time is 12 s for visiting 1040 nodes, and the CL-AtSe verification analyzes 3 states with 0.13 s to translate. Because the summary part of OFMC and CL-AtSe indicates that the protocol is SAFE, our proposed protocol is secure against replay and man-in-the-middle attacks.

```
role gateway(UA, GA, CS : agent, SKgacs : symmetric_key,
 H: hash_func, SND, RCV : channel(dy))

played_by GA

def=
local State: nat,
    IDi,PWi,BIOi,Ri,Pi,HIDi, HPWi,GIDj,Li, Ai, UIi, Aii, Bi : text,
    Ni,Nj,Nc, Ci, Xi, X, Y, Bj,VUi: text,
    Di,Ei,Fi,Gi,Hi, VSj, Mcg,Mcu, GIj, PIDj : text,
    HIDinew, UIinew, Aiinew, Binew, Xinew : text,
    SKi, SKj, SKc: text

const sp1, sp2, sp3, sp4 : protocol_id,
 ua_cs_ni, cs_ua_mcu, ga_cs_nj, cs_ga_mcg: protocol_id
init State := 0
transition

1. State = 0 /\ RCV(start) =|>
  State' := 1 /\ Bj' := new()
          /\ SND({GIDj.Bj'}_SKgacs)
          /\ RCV({H(GIDj.Bj').H(H(GIDj.Bj').H(X.Y))}_SKgacs)
          /\ secret({Bj},sp2,{GA,CS})

2. State = 2
/\ RCV(H(IDi.Ai').xor(H(H(IDi.PWi.Ai').Ai'.X),Ni').H(H(H(IDi.Ai').Ai'.X).X).Ni'.
GIDj)) =|>

State' := 3 /\ Nj':= new() /\ Bj' := new()
        /\ Di' := xor(H(H(GIDj.Bj').H(X.Y)),Nj')
        /\ VSj' := H(GIDj.H(H(GIDj.Bj').H(X.Y)).Nj')
        /\ SND(H(IDi.Ai').xor(H(H(IDi.Ai').Ai'.X),Ni').H(H(H(IDi.Ai').Ai'.X).X).Ni'.
GIDj).H(GIDj.Bj').Di'.VSj')
        /\ witness(GA,CS,ga_cs_nj,Nj')

3. State = 3
/\ RCV(H(xor(H(GIDj'.Ni'),Nc').H(GIDj.Ni').Nc').H(H(H(H(IDi.Ai').Ai'.X).X).H(H(I
Di.Ai').Ai'.X).Ni').xor(H(GIDj.Ni),Nc').xor(H(GIDj.Ni),Ni')) =|>

State' := 4 /\ SKj' :=  H(xor(Ni',H(Nj'.Nc')))/\ Nj':= new()
        /\ Gi' := H(GIDj.Ni')
        /\ Hi' := xor(Gi',H(Nj'.Nc'))
        /\ SND(Hi'.H(H(H(H(IDi.Ai').Ai'.X).X).H(H(IDi.Ai').Ai'.X).Ni'))
        /\ request(GA,CS,cs_ga_mcg,Nc')

end role
```

Figure 10. Specification of gateway.

```
role controlserver(UA, GA, CS : agent, SKuacs, SKgacs : symmetric_key, H:
hash_func, SND, RCV : channel(dy))

played_by CS
def=
local State: nat,
   IDi,PWi,BIOi,Ri,Pi,HIDi, HPWi,GIDj,Li, Ai, UIi, Aii, Bi, Ni,Nj,Nc, Ci, Xi, X, Y,
Bj,VUi: text,
   Di,Ei,Fi,Gi,Hi, VSj, Mcg,Mcu, GIj, PIDj : text,
   HIDinew, UIinew, Aiinew, Binew, Xinew : text,
   SKi, SKj, SKc: text
const sp1, sp2, sp3, sp4, ua_cs_ni, cs_ua_mcu, ga_cs_nj, cs_ga_mcg : protocol_id
init State := 0
transition

1. State = 0 /\ RCV({GIDj.Bj'}_SKgacs) =|>
State' := 1 /\ PIDj' := H(GIDj.Bj')
     /\ GIj' := H(H(GIDj.Bj').H(X.Y))
     /\ SND({PIDj'.GIj'}_SKgacs)
     /\ secret({X,Y},sp3,{CS})
     /\ secret({PIDj',GIj'},sp4,{GA,CS})
2. State = 1 /\ RCV({H(IDi.Ai').H(IDi.PWi.Ai')}_SKuacs) =|>
State' := 2 /\ UIi' := H(H(IDi.Ai').Ai'.X)
     /\ Aii' := xor(UIi,H(H(IDi.PWi.Ai')))
     /\ Bi' := H(UIi'.Aii')
     /\ Xi' := H(UIi'.X)
     /\ SND({Aii'.Bi'.Xi'}_SKuacs)
3. State = 2
/\ RCV(H(IDi.Ai').xor(H(H(IDi.Ai').Ai'.X),Ni').H(H(H(H(IDi.Ai').Ai'.X).X).Ni'.GIDj
).H(GIDj'.Bj).xor(H(H(GIDj.Bj').H(X.Y)),Nj').H(GIDj.H(H(GIDj.Bj').H(X.Y)).Nj'))
=|>
State' := 3 /\ Nc' := new()
     /\ Ei' := xor(H(GIDj.Ni'),Nc')
     /\ Fi' := xor(H(GIDj.Ni'),Ni')
     /\ Mcg' := H(xor(H(GIDj.Ni'),Nc').H(GIDj.Ni').Nc')
     /\ SKc' := H(xor(Ni',H(Nj'.Nc')))
      /\ Mcu' := H(H(H(H(IDi.Ai').Ai'.X).X).H(H(IDi.Ai').Ai'.X).Ni')
     /\ HIDinew' := H(H(IDi.Ai').Ni'.H(Nj'.Nc'))
     /\ UIinew' := H(HIDinew'.Ni'.H(H(IDi.Ai').Ai'.X))
     /\ witness(CS,UA,cs_ua_mcu,Nc')
     /\ witness(CS,GA,cs_ga_mcg,Nc')
     /\ SND(Mcg'.Mcu'.Ei'.Fi')
     /\ request(UA,CS, ua_cs_ni,Ni')
     /\ request(GA,CS, ga_cs_nj,Nj')


end role
```

Figure 11. Specification of control server.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/APMI.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 12.00s
  visitedNodes: 1040 nodes
  depth: 9 plies
```

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/APMI.if

GOAL
  As Specified

BACKEND
  CL-AtSe

STATISTICS

  Analysed   : 3 states
  Reachable  : 3 states
  Translation: 0.13 seconds
  Computation: 0.00 seconds
```

Figure 12. The result of Automated Validation of Internet Security Protocols and Applications (AVISPA) simulation using OFMC and CL-AtSe.

9. Performance Analysis

In this section, we show the comparison of computation cost, communication cost, and security features among our proposed protocol and other IoT-related protocols.

9.1. Computation Cost

We compare the computational overhead between our proposed protocol and other related protocols. We define some notations for convenience of comparison.

- T_{me}: The times for modular exponential operation (≈0.522 s [33,34])
- T_h: The times for one-way hash operation (≈0.0005 s [33,34])
- T_f: The times for fuzzy extraction operation (≈0.063075 s [34,35])

Table 4 shows the results of the comparison. In multi-gateway environments, it is important to reduce the computation cost of gateway nodes because the gateway nodes process a large amount of information. Although the total computation cost of our proposed protocol is higher than other related protocols, it is similar to [15] in terms of gateway nodes. Therefore, our proposed protocol is suitable for practical IoT environments.

Table 4. Computation cost of the login and authentication phase.

Protocols	User	Gateway	Control Server	Total Cost
Turkanovic et al. [5]	$7T_h$	$5T_h$	$7T_h$	$19T_h(0.0095s)$
Wu et al. [3]	$2T_{me} + 4T_h$	-	$1\ T_{me} + 4T_h$	$3T_{me}+ 8T_h(1.57s)$
Amin and Biswas Case-1 [10]	$7T_h$	$5T_h$	$8T_h$	$20T_h(0.01s)$
Amin and Biswas Case-2 [10]	$8T_h$	$5T_h$	$7T_h$	$20T_h(0.01s)$
Bae et al. [15]	$5T_h$	$6T_h$	$10T_h$	$21T_h(0.0105s)$
Ours	$1T_f+14T_h$	$5T_h$	$9T_h$	$1T_f+ 28T_h(0.07707s)$

XOR operation is negligible compared to other operations.

9.2. Communication Cost

We have compared the communication overheads at the login and authentication phase of our proposed protocol and other related protocols in Table 5. We assume that the acknowledgment message and the one-way hash function, the timestamp, random number, and identity all are 160 bits. Additionally, we assume that the AES (Advanced Encryption Standard) key is 512 bits [33]. According to the results, our proposed protocol has more efficiency than other related protocols.

Table 5. Communication cost.

Protocols	Communication Cost
Turkanovic et al. [5]	4000 bits
Wu et al. [3]	2368 bits
Amin and Biswas Case-1 [10]	2080 bits
Amin and Biswas Case-2 [10]	3520 bits
Bae et al. [15]	2720 bits
Ours	2400 bits

9.3. Security Properties

Table 6 shows the security comparisons among the proposed protocol and other related protocols based on IoT environment. Our proposed protocol can resist more attacks than other related protocols. Furthermore, our proposed protocol provides anonymity and achieves mutual authentication. Therefore, we demonstrate that the proposed protocol is more safe than other related protocols and satisfies the security requirements of IoT environments.

Table 6. Security properties.

Security Property	Turkanovic et al. [5]	Wu et al. [3]	Amin and Biswas [10]	Bae et al. [15]	Ours
User impersonation attack	x	x	o	x	o
Server spoofing attack	o	x	x	x	o
Smartcard stolen attack	x	x	x	x	o
Trace attack	x	x	x	x	o
Off-line password guessing attack	x	o	x	o	o
Replay attack	o	o	o	o	o
Man-in-the-middle attack	o	o	o	o	o
Desynchronization attack	-	-	x	-	o
Anonymity	x	x	x	o	o
Mutual authentication	x	x	o	x	o

x: does not prevent the property; o: prevents the property; -: does not concern the property.

10. Conclusions

IoT is becoming a part of our life and helps people to easily communicate data and comfortably obtain mobile services. However, data scalability, unsolved security problems, and malicious attacks can limit the widespread extension of IoT services. The gateway nodes must process a large amount of information to provide IoT services to users. Thus, reducing the computation cost of gateways is a very important issue, and users and gateways should verify each other's legitimacy with the aid of a control server to provide authorized and secure communication. In this paper, we demonstrated the security weaknesses of Bae et al.'s protocol. We showed that their protocol is vulnerable to user impersonation attacks, gateway spoofing attacks, session key disclosure attacks, offline password guessing attacks, and does not provide secure mutual authentication. Moreover, we proposed a multi-factor mutual authentication protocol for multi-gateway IoT environments with better security functionality than that of Bae et al.'s protocol. We also proved the security of the proposed protocol using BAN logic and the AVISPA tool.

Author Contributions: Y.P. (YoungHo Park) supervised the research and contributed to manuscript organization; J.L. and S.Y. contributed to the protocol design; K.P. and Y.P. (YoHan Park) analyzed and revised the manuscript; J.L., S.Y., and K.P. performed the experiments and analyzed the protocol. J.L. and Y.P. (YoHan Park) wrote the manuscript. All authors took part in paper preparation and edition.

Funding: This work was supported by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Science, ICT, and Future Planning under Grant 2017R1A2B1002147 and in part by the BK21 Plus project funded by the Ministry of Education, Korea, under Grant 21A20131600011.

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

References

1. Wu, F.; Xu, L.; Kumari, S.; Li, X.; Shen, J.; Choo, K.R.; Wazid, M.; Das, A.K. An efficient authentication and key agreement scheme for multi-gateway wireless sensor networks in IoT deployment. *J. Netw. Comput. Appl.* **2017**, *81*, 72–85. [CrossRef]
2. Das, A.K.; Sutrala, A.K.; Kumari, S.; Odelu, V.; Wazid, M.; Li, X. An efficient multi-gateway-based three-factor user authentication and key agreement scheme in hierarchical wireless sensor networks. *Secur. Commun. Netw.* **2016**, *9*, 2070–2092. [CrossRef]
3. Wu, Z.Y.; Lee, Y.C.; Lai, F.; Lee, H.C.; Chung, Y. A secure authentication scheme for telecare medicine information systems. *J. Med. Syst.* **2010**, *36*, 1529–1535. [CrossRef]
4. Chang, Y.F.; Yu, S.H.; Shiao, D.R. A uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care. *J. Med. Syst.* **2013**, *37*, 9902. [CrossRef] [PubMed]
5. Turkanović, M.; Brumen, B.; Hölbl, M. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion. *Ad Hoc Netw.* **2014**, *20*, 96–112. [CrossRef]
6. He, D.; Chen, J.; Zhang, R. A more secure authentication scheme for telecare medicine information systems. *J. Med. Syst.* **2012**, *36*, 1989–1995.
7. Das, A.K.; Goswami, A. A secure and efficient uniqueness-and-anonymity-preserving remote user authentication scheme for connected health care. *J. Med. Syst.* **2013**, *37*, 9948. [CrossRef] [PubMed]
8. Kumari, S.; Li, X.; Wu, F.; Das, A.K.; Choo, K.; Shen, J. Design of a provably secure biometrics-based multi-cloud-server authentication scheme. *Future Gener. Comput. Syst.* **2017**, *68*, 320–330. [CrossRef]
9. Chatterjee, S.; Roy, S.; Das, A.K.; Chattopadhyay, S.; Kumar, N. Secure biometric-based authentication scheme using chebyshev chaotic map for multi-server environment. *IEEE Trans. Depend. Sec. Comput.* **2018**, *15*, 428–442. [CrossRef]
10. Amin, R.; Biswas, G.P. A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. *Ad Hoc Netw.* **2016**, *36*, 58–80. [CrossRef]
11. Lin, C.H.; Lai, Y.Y. A flexible biometrics remote user authentication scheme. *Comput. Stand. Interfaces* **2004**, *27*, 19–23. [CrossRef]
12. Dhillon, P.K.; Kalra, S. A lightweight biometrics based remote user authentication scheme for IoT services. *J. Inf. Secur. Appl.* **2017**, *34*, 255–270. [CrossRef]
13. Srinivas, J.; Mukhopadhyay, S.; Mishra, D. Secure and efficient user authentication scheme for multi-gateway wireless sensor networks. *Ad Hoc Netw.* **2017**, *54*, 147–169. [CrossRef]
14. Wu, F.; Li, X.; Sangaiah, A.K.; Xu, L.; Kumari, S.; Wu, L.; Shen, J. A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks. *Future Gener. Comput. Syst.* **2018**, *82*, 727–737. [CrossRef]
15. Bae, W.; Kwak, J. Smart card-based secure authentication protocol in multi-server IoT environment. *Multimed. Tools. Appl.* **2017**, 1–19. [CrossRef]
16. Xu, G.; Qiu, S.; Ahmad, H.; Xu, G.; Guo, Y.; Zhang, M.; Xu, H. A multi-server two-factor authentication scheme with un-traceability using elliptic curve cryptography. *Sensors* **2018**, *18*, 2394. [CrossRef] [PubMed]
17. Leu, J.S.; Chen, C.F.; Hsu, K.C. Improving heterogeneous SOA-based IoT message stability by shortest processing time scheduling. *IEEE Trans. Serv. Comput.* **2013**, *99*, 1–99. [CrossRef]
18. Dolev, D.; Yao, A.C. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
19. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology*; Springer Science+Business Media: Berlin, Germany; New York, NY, USA, 1999; pp. 388–397.
20. Park, Y.; Park, Y. Three-factor user authentication and key agreement using elliptic curve cryptosystem in wireless sensor networks. *Sensors* **2016**, *16*, 2123. [CrossRef]

21. Burnett, A.; Byrne, F.; Dowling, T.; Duffy, A. A biometric identity based signature scheme. *Int. J. Netw. Secur.* **2007**, *5*, 317–326.
22. Dodis, Y.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *Proc. Adv. Cryptol.* **2004**, *3027*, 523–540.
23. Park, Y.; Park, K.; Lee, K.; Song, H.; Park, Y. Security analysis and enhancements of an improved multi-factor biometric authentication scheme. *Int. J. Distrib. Sens. Netw.* **2017**, *13*, 1–12. [CrossRef]
24. Chatterjee, S.; Roy, S.; Das, A.K.; Chattopadhyay, S.; Kumar, N.; Alavalapati, G.R.; Park, K.; Park, Y.; Park, Y. On the design of fine grained access control with user authentication scheme for telecare medicine information systems. *IEEE Access* **2017**, *5*, 7012–7030. [CrossRef]
25. Von Oheimb, D. The high-level protocol specification language HLPSL developed in the EU project avispa. In Proceedings of the APPSEM 2005 Workshop, Tallinn, Finland, 13–15 September 2005; pp. 1–2.
26. Park, K.; Park, Y.; Park, Y.; Reddy, A.G.; Das, A.K. Provably secure and efficient authentication protocol for roaming service in global mobility networks. *IEEE Access* **2017**, *5*, 25110–25125. [CrossRef]
27. Park, K.; Park, Y.; Park, Y.; Das, A.K. 2PAKEP: Provably secure and efficient two-party authenticated key exchange protocol for mobile environment. *IEEE Access* **2018**, *6*, 30225–30241. [CrossRef]
28. 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]
29. Turuani, M. The CL-Atse protocol analyser. In Proceedings of the International Conference on Rewriting Techniques and Applications (RTA), Seattle, WA, USA, 12–14 August 2006; pp. 227–286.
30. Basin, D.; Modersheim, S.; Vigano, L. OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Secur.* **2005**, *4*, 181–208. [CrossRef]
31. AVISPA. Automated Validation of Internet Security Protocols and Applications. Available online: http://www.avispa-project.org/ (accessed on 11 January 2019).
32. SPAN: A Security Protocol Animator for AVISPA. Available online: http://www.avispa-project.org/ (accessed on 11 January 2019).
33. Preeti, C.; Hari, O. A secure and robuts anonymous three-factor remote user authentication scheme for multi-server environment using ECC. *Comput. Commun.* **2017**, *110*, 26–34.
34. Li, C.; Hwang, M.S.; Chu, Y.P. A secure and efficient communication scheme with authenticated key establishment and privacy preserving for vehicular ad hoc networks. *Comput. Commun.* **2008**, *31*, 2803–2814. [CrossRef]
35. Ostad-Sharif, A.; Abbasinezhad-Mood, D.; Nikooghadm, M. A robust and efficient ECC-based mutual authentication and session key generation scheme for healthcare applications. *J. Med. Syst.* **2019**, *43*, 1–22. [CrossRef]