An Enhanced Lightweight IoT-based Authentication Scheme in Cloud Computing Circumstances

Rafael Martínez-Peláez [1,*] , **Homero Toral-Cruz** [2], **Jorge R. Parra-Michel** [1] , **Vicente García** [3] , **Luis J. Mena** [4], **Vanessa G. Félix** [4] and **Alberto Ochoa-Brust** [5]

[1] Facultad de Tecnologías de Información, Universidad De La Salle Bajío, Av. Universidad 602, León 37150, Mexico; jrparra@delasalle.edu.mx

[2] Department of Sciences and Engineering, University of Quintana Roo, Blvd Bahía S/N, Chetumal 77019, Mexico; htoral@uqroo.edu.mx

[3] Departamento de Ingeniería Eléctrica y Computación, Universidad Autónoma de Ciudad Juárez, Av. José de Jesús Macías Delgado 18100, Cd. Juárez 32310, Mexico; vicente.jimenez@uacj.mx

[4] Unidad Académica de Computación, Universidad Politécnica de Sinaloa, Ctra. Libre Mazatlán Higueras Km 3, Mazatlán 82199, Mexico; lmena@upsin.edu.mx (L.J.M.); vfelix@upsin.edu.mx (V.G.F.)

[5] Facultad de Ingeniería Mecánica y Eléctrica, Universidad de Colima, Av. Universidad 333, Colima 28040, Mexico; aochoa@ucol.mx

* Correspondence: rmartinezp@delasalle.edu.mx; Tel.: +52-477-710-8567

check for updates

Abstract: With the rapid deployment of the Internet of Things and cloud computing, it is necessary to enhance authentication protocols to reduce attacks and security vulnerabilities which affect the correct performance of applications. In 2019 a new lightweight IoT-based authentication scheme in cloud computing circumstances was proposed. According to the authors, their protocol is secure and resists very well-known attacks. However, when we evaluated the protocol we found some security vulnerabilities and drawbacks, making the scheme insecure. Therefore, we propose a new version considering login, mutual authentication and key agreement phases to enhance the security. Moreover, we include a sub-phase called evidence of connection attempt which provides proof about the participation of the user and the server. The new scheme achieves the security requirements and resists very well-known attacks, improving previous works. In addition, the performance evaluation demonstrates that the new scheme requires less communication-cost than previous authentication protocols during the registration and login phases.

1. Introduction

Information technology has grown rapidly in the past few years, mainly developing new technologies focused on how to take advantage of the Internet. In this sense, innovations such as wireless access, high speed connection, APIs and electronic services have successfully entered the Internet arena. At the same time, researchers and computer professionals have developed new communication technologies, such as Wi-Fi, 4G, 5G, routing protocols, LTE, Bluetooth, RFID, among others, which offer broader ranges to users. In parallel, the cost of technology keeps decreasing, year by year, making Internet connectivity more accessible for everyone through smaller devices as tablets and smartphones.

On the other hand, as the Internet has become ubiquitous, faster, and increasingly accessible to non-technical communities, social networking and collaborative services, emerging technologies such as artificial intelligence, big data, cloud computing, wireless sensor networks and the Internet of Things (IoT) have appeared, enabling people to communicate and share interests in many more ways.

As a consequence, these novel technologies are changing the world, again, creating new business opportunities, new applications to enhance safety, comfort, and efficiency reducing human efforts, and new ways to collect and analyse data.

Among these five emerging technologies, the IoT and cloud computing have become more and more relevant to academia and industry. In 1999 Ashton introduced the concept of IoT [1], which is defined as the connection of physical objects (devices/sensors) through the Internet [2]. Cloud computing, on the other hand, was introduced in 1961 by McCarthy [3], and 36 years later, Chellappa explained the concept in the scenario of current technology era [4], defined as a large-scale distributed computing paradigm which drives the economy of many companies based on virtualization, managed computing power, and storage focused on its core business [3].

The IoT applications are classified into the following categories [5,6]: (a) Internet of sensors (IoS), which is a network made up of sensors which collect and transmit a types of data; (b) Internet of energy (IoE), which is a network of smart grids to analyse and control the energy production, consumption, storage, and distribution; (c) machine to machine (M2M) communication is a network of devices/sensors connected through the Internet; and (d) Internet of Vehicles (IoV), which is a network of vehicles that can share information about the state of the road. Another classification is based on the communication models [7]: (a) machine-to-machine communication, which includes multiple devices/sensors connected to exchange data among them without a physical infrastructure; (b) machine-to-cloud communication, which includes devices/sensors that consume services from a cloud server for storing and processing data; and (c) machine-to-gateway communication, which includes devices/sensors that collaborate as a proxy to expand the range of the network.

In the case of cloud computing, we can find four types of deployment models offered by cloud providers. The first type is known as public cloud, where services and systems are available for anyone through an open communication channel. The second type is known as private cloud where services and systems are accessible for certain users/employees or organizations through a secure communication channel. The third type is known as community cloud where the cloud is shared by several organizations. The last type which includes a public and private cloud to share resources is known as hybrid cloud [4].

From these service platforms, cloud computing allows the interconnection of everything around us, including our vital signs, personal and sensitive data, that travel through an open communication channel to different sites [8–10]. This is possible because devices/sensors automatically collect a huge amount of data and stores them on cloud servers [11–14]. However, this also represents a significant disadvantage in terms of security, because large amounts of personal and sensitive data stored in a single database could be accessed without the approval of users [9].

On the other hand, the main advantage of the IoT, ubiquity, is also its main weakness, because it is also necessary to have a high and complex security protocol. According to El-Hajj et al. [6] and Ferrag et al. [5], IoT security requirements include authentication, authorization, integrity, confidentiality, non-repudiation, availability, and privacy to protect the data, nodes, and messages against attacks.

Therefore, to address the security requirements of both emerging technologies is important focus special attention on the authentication process, because it is the first line of defense against potential attackers. The goal of the authentication protocol is verifying the identity of an entity to determine that he/she or device/sensor is who or what it claims to be [5,6]. For this reason, the authentication process is a key component for secure Internet communication.

In this sense, a strong authentication protocol needs to achieve the following two aims: mutual authentication and session key agreement [15]. In addition, the authentication protocol must avoid the denial-of-service, forgery, parallel session, password guessing, replay, smart card loss, and stolen-verifier attacks [15]. In the threat model of any authentication protocol, an adversary or malicious user has the computational power to compute complex operations in low time and control the public communication channel to capture and store any messages. In general, security threats include more than thirty kinds

of attacks [5]. However, the most popular attacks used for evaluating authentication protocols are man-in-the-middle, impersonation and forging, and replay attacks [5].

In recent years, authentication protocols have used cryptosystems as countermeasures to enhance security [5]. The main cryptosystems are hash functions, symmetric algorithms (AES), asymmetric algorithms (RSA, D-H, ECC), digital signatures, and ID-based cryptography; and its adoption mainly depends on the deployment, computational power and energy consumption of the device/sensor. Thus, cryptosystems that require more computational power must be implemented in device/sensor that needs high energy consumption. Therefore, we address to lightweight authentication protocols, which require low computational operations. In this way, the studies carried out by Wang et al. in 2015 [16] and 2018 [17] contribute to understanding the security requirements, adversary model, types of schemes, and evaluation criteria of lightweight authentication protocols.

1.1. Related Work

In this paper, we refer to lightweight authentication schemes which require low computational operations, such as hash functions and exclusive-OR operation. The first lightweight authentication scheme was proposed by Lamport in 1981 [18]. Lamport introduced the concept of a hash chain to authenticate remote users through an open communication channel. Lamport's scheme is feasible for practical implementation due to its low computational cost, however, the scheme requires that the server maintains a verification table making it vulnerable to steal personal data. In 1990, Hwang et al. [19] proposed a scheme without a verification table. Later, Lin et al. [20] proposed an authentication scheme based on the asymmetric ElGamal algorithm to improve the security characteristics of Li et al. in [21]. Since 1990, several authentication schemes were proposed to enhance the security of previous ones. These schemes were designed for communication between *n* users and a single server.

Later, Liao et al. [22] proposed an authentication scheme for a multi-server environment. Nevertheless, Liao et al.'s scheme is vulnerable to an insider attack, masquerade attack, server spoofing attack, and registration center spoofing attacks [23]. Hsiang et al. [23] proposed a new authentication scheme to resolve the security drawbacks of Liao et al.'s scheme. However, Martínez-Peláez et al. [24] demonstrated that Hsiang et al.'s scheme is insecure. Two years later, Kim et al. [25] evaluated Martínez-Peláez et al.'s scheme finding security vulnerabilities. The same year, Li et al. [26] evaluated the scheme proposed by Sood et al. [27] finding it insecure. Then, Xue et al. [28] demonstrated the security vulnerabilities of Li et al.'s scheme. Later, Amin et al. [1] proposed an authentication scheme which remedies the security drawbacks of Xue et al.'s scheme. Nonetheless, Challa et al. [29] demonstrated that Amin et al.'s scheme is vulnerable to privileged-insider and impersonation attacks.

In 2019, Zhou et al. [30] proposed a scheme based on hash function and exclusive-or operation to provide authentication on large-scale IoT and cloud computing deployment. They explained that Amin et al.'s scheme cannot resist off-line guessing attacks. They also claimed that their scheme is secure against very well-known attacks.

1.2. Contribution

In this work, we review the scheme proposed by Zhou et al. [30] and point out that the scheme has security vulnerabilities and drawbacks which make it insecure. In particular, the scheme is vulnerable to insider, replay and user impersonation attacks. Moreover, the scheme fails to provide mutual authentication and fails to protect secret keys. Therefore, the main contribution of this paper is a new version of the authentication scheme proposed by Zhou et al. which achieves the following security characteristics: mutual authentication and session key agreement. Moreover, our proposal maintains the user's anonymity against eavesdroppers and requires login phase.

On the other hand, in the security scheme of Zhou et al., neither the server or nor the user knows if the other party is a legal member of the system. For that reason, the scheme includes a new sub-phase called evidence of connection attempt which provides elements for identifying the participants of the authentication request. This sub-phase complements the authentication phase included in [30].

The rest of this paper is organized as follows: Section 2 presents the overview of Zhou et al.'s scheme, in particular registration and authentication phases, and contains the results of the security analysis carried out to the proposal of Zhou et al. based on [5,6,15]. The new proposal is explained in Section 3. Security analysis and performance evaluation are given in Section 4. Conclusions are presented in Section 5.

2. Review and Security Analysis of Zhou's Scheme

Firstly, we provide a brief description about the registration and authentication phases of the scheme proposed by Zhou et al. in [30]. Then, we carry out the security analysis to explain the drawbacks and vulnerabilities found in their scheme.

2.1. Registration Phase

This phase is divided in user registration and cloud server registration sub-phases.

2.1.1. User Registration Sub-Phase

In this sub-phase, the user (U_i) is registered by the control server (CS). The communication between U_i and CS is through a secure channel. The steps involved in this sub-phase are as follows:

Step 1: U_i selects an identity (ID_i), pseudo-identity (PID_i), password (PW_i), and nonce (b_i). Then, U_i computes $HP_i = h(PW_i \parallel b_i)$ and sends the registration request message $M_1 = \{ID_i, PID_i\}$ to CS were $h(\cdot)$ is a one-way hash function, $\parallel$ represents concatenation operation and $\oplus$ represents an exclusive-or operation.

Step 2: Upon receiving the registration request message M_1, CS verifies if ID_i is valid or not. In case that ID_i is invalid, the registration process will be closed. On the other hand, CS computes $C_1^* = h(PID_i \parallel ID_{CS} \parallel x)$ and $C_2^* = h(ID_i \parallel x)$. Then, CS stores ID_i in a database and sends the registration response message $M_2 = \{C_1, C_2, ID_{CS}\}$ to U_i.

Step 3: After receiving the registration response message M_2, U_i computes $C_1 = C_1^* \oplus HP_i$, $C_2 = C_2^* \oplus h(ID_i \parallel HP_i)$ and $C_3 = b_i \oplus h(ID_i \parallel PW_i)$, and stores $(C_1, C_2, C_3, PID_i, ID_{CS})$ in his smart card.

At this point, the user registration sub-phase is over and U_i was registered by CS.

2.1.2. Cloud Server Registration Sub-phase

In this sub-phase, the server (S_j) is registered by the control server (CS). The communication between S_j and CS is through a secure communication channel. The steps involved in this sub-phase are as follows:

Step 1: S_j selects an identity (SID_j) and pseudo-identity $(PSID_j)$. Then, S_j sends the registration request message $M_3 = \{SID_j, PSID_j\}$ to CS.

Step 2: Upon receiving the registration request message M_3, CS computes $B_1 = h(PSID_j \parallel ID_{CS} \parallel x)$ and $B_2 = h(SID_j \parallel x)$. Then, CS stores SID_j and sends the registration response message $M_4 = \{B_1, B_2, ID_{CS}\}$ to S_j.

Step 3: After S_j receives the registration response message M_4, S_j stores $(B_1, B_2, SID_j, PSID_j, ID_{CS})$.

At this point, the cloud server registration sub-phase is over and S_j was registered by CS.

2.2. Authentication Phase

This phase is divided in the following steps and the details are shown in Figure 1:

Step 1: This step is invoked by U_i, when she/he wants to get access to the service offered by S_j. U_i inserts his/her smart card and keys his/her ID_i and PW_i. Then, the smart card generates a random number (r_U) and new pseudo-identity $\left(PID_i^{new}\right)$. After that, U_i computes $b_i = C_3 \oplus h(ID_i \parallel PW_i)$,

$HP_i = h(PW_i \parallel b_i)$, $C_1^* = C_1 \oplus HP_i$, $C_2^* = C_2 \oplus h(ID_I \parallel HP_I)$, $D_1 = C_1^* \oplus r_U$, $D_2 = h(PID_i \parallel ID_{CS} \parallel r_U) \oplus ID_i$, $D_3 = C_2^* \oplus h(ID_i \parallel HP_i) \oplus PID^{new} \oplus h(r_U \parallel ID_i)$, and $D_4 = h\left(ID_i \parallel PID_i \parallel PID_i^{new} \parallel r_U \parallel D_3\right)$. Then, U_i sends the authentication request message $M_5 = \{PID_i, D_1, D_2, D_3, D_4\}$ to S_j through an open communication channel.
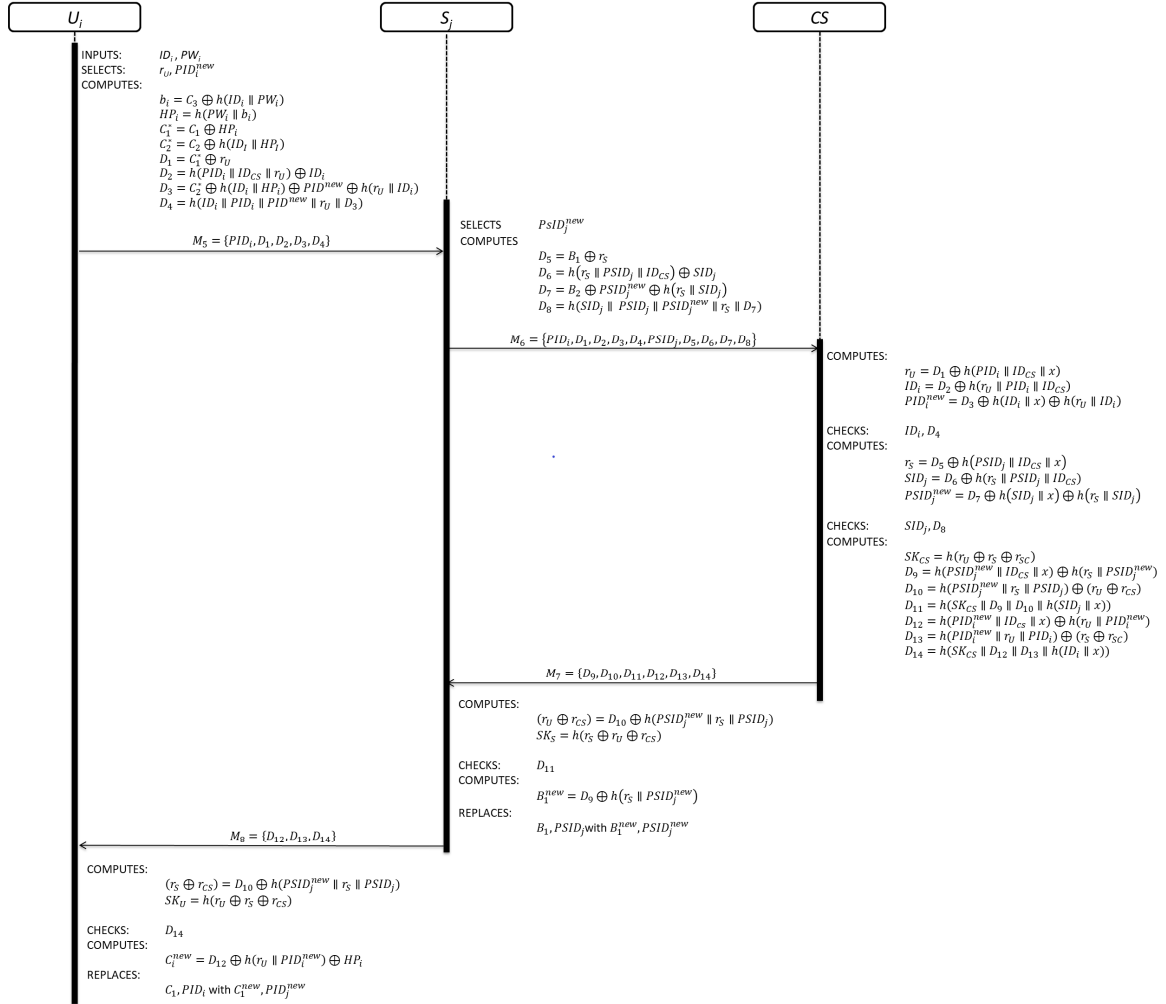

Figure 1. Authentication phase of Zhou et al.'s scheme.

Step 2: This step is invoked by S_j. After receiving the authentication request message M_5, S_j selects a new pseudo-identity $\left(PSID_j^{new}\right)$ and a random number (r_S). Then, S_j computes $D_5 = B_1 \oplus r_S$, $D_6 = h\left(r_S \parallel PSID_j \parallel ID_{CS}\right) \oplus SID_j$, $D_7 = B_2 \oplus PSID_j^{new} \oplus h\left(r_S \parallel SID_j\right)$, and $D_8 = h\left(SID_j \parallel PSID_j \parallel PSID_j^{new} \parallel r_S \parallel D_7\right)$. Finally, S_j sends the authentication request message $M_6 = \left\{PID_i, D_1, D_2, D_3, D_4, PSID_j, D_5, D_6, D_7, D_8\right\}$ to CS through an open communication channel.

Step 3: This step is invoked by CS. Upon receiving the authentication request message M_6, CS computes $r_U = D_1 \oplus h(PID_i \parallel ID_{CS} \parallel x)$, $ID_i = D_2 \oplus h(r_U \parallel PID_i \parallel ID_{CS})$, and $PID_i^{new} = D_3 \oplus h(ID_i \parallel x) \oplus h(r_U \parallel ID_i)$. Then, CS checks if ID_i and D_4 are valid or not. If the verification process fails, CS closes the communication. On the other hand, CS computes $r_S = D_5 \oplus h\left(PSID_j \parallel ID_{CS} \parallel x\right)$, $SID_j = D_6 \oplus h\left(r_S \parallel PSID_j \parallel ID_{CS}\right)$, and $PSID_j^{new} = D_7 \oplus h\left(SID_j \parallel x\right) \oplus h\left(r_S \parallel SID_j\right)$. Next, CS checks if SID_j and D_8 are valid or not. If the verification process fails, CS closes the communication. On the other hand, CS selects a random number (r_{CS}) and computes the session key $SK_{CS} = h(r_U \oplus r_S \oplus r_{SC})$. Later, CS computes $D_9 = h\left(PSID_j^{new} \parallel ID_{CS} \parallel x\right) \oplus$

$h\left(r_S \parallel PSID_j^{new}\right),$ $D_{10} = h\left(PSID_j^{new} \parallel r_S \parallel PSID_j\right) \oplus (r_U \oplus r_{CS}),$ $= h\left(SK_{CS} \parallel D_9 \parallel D_{10} \parallel h\left(SID_j \parallel x\right)\right),$ $D_{12} = h\left(PID_i^{new} \parallel ID_{cs} \parallel x\right) \oplus h\left(r_U \parallel PID_i^{new}\right),$ $D_{13} = h\left(PID_i^{new} \parallel r_U \parallel PID_i\right) \oplus (r_S \oplus r_{SC}),$ and $D_{14} = h(SK_{CS} \parallel D_{12} \parallel D_{13} \parallel h(ID_i \parallel x))$. Finally, CS sends $M_7 = \{D_9, D_{10}, D_{11}, D_{12}, D_{13}, D_{14}\}$ to S_j through an open communication channel.

Step 4: This step is invoked by S_j. Yet receiving the authentication response message M_7, S_j computes $(r_U \oplus r_{CS}) = D_{10} \oplus h\left(PSID_j^{new} \parallel r_S \parallel PSID_j\right)$ and $SK_S = h(r_S \oplus r_U \oplus r_{CS})$. Then, S_j checks if D_{11} is correct or not. If the verification process is correct, S_j computes $B_1^{new} = D_9 \oplus h\left(r_S \parallel PSID_j^{new}\right)$ and replaces B_1, $PSID_j$ with B_1^{new}, $PSID_j^{new}$. Finally, S_j sends $M_8 = \{D_{12}, D_{13}, D_{14}\}$ to U_i through an open communication channel.

Step 5: This step is invoked by U_i. After receiving the authentication response message M_8, U_i computes $(r_S \oplus r_{SC}) = D_{13} \oplus h\left(PID_i^{new} \parallel r_U \parallel PID_i\right)$ and $SK_U = h(r_U \oplus r_S \oplus r_{CS})$. Then, U_i checks if D_{14} is correct or not. If the verification process is correct, U_i computes $C_i^{new} = D_{12} \oplus h\left(r_U \parallel PID_i^{new}\right) \oplus HP_i$ and replaces C_1, PID_i with C_1^{new}, PID_i^{new}.

2.3. Security Vulnerabilities

In this subsection, we explain the weaknesses found in the scheme proposed by Zhou et al. in [30]. The security analysis was conducted based on [5,6,15]. From these works, we performed the security analysis. In this part, we assume the following capabilities of the attacker [31,32]:

- The attacker is a legal member of the system which means that he/she was registered by CS and he/she has all the security parameters.
- The attacker can control the public communication channel giving him/her the possibility to intercept, insert, store, delete, or modify any message.
- The attacker has high computational power connected to the public communication channel.

2.3.1. Insider Attack

This attack happens when a malicious user has enough knowledge to attack sensitive data or the whole system. In this scenario, the attacker knows $C_2^* = h(ID_i \parallel x)$, computed by CS during the user registration phase. According with Zhou et al., CS uses the same secret key (x) to register users and servers, so he/she can find x from C_2^*, searching exhaustively all possible random number y until $h(ID_i \parallel y) = C_2^* = h(ID_i \parallel x)$ to know the secret key of CS. This attack is possible because the attacker knows ID_i and C_2^*.

2.3.2. Man-in-the-Middle Attack

This attack happens when an attacker has control over the public communication channel providing him/her the possibility to listen the conversation between two entities. Under this scenario, the attacker can intercept the authentication request message $M_5 = \{PID_i, D_1, D_2, D_3, D_4\}$ sent from U_i to S_j. Under this situation, the attacker can achieve the following active attacks.

Replay Attack

The attacker can transmit the last authentication request message M_5 sent to S_j, at any given time and from any given place, to impersonate a legitimate U_i. The description of the attack is as follows:

Step 1: The attacker has, at-least, one authentication request message M_5 sent by U_i to S_j, and the attacker knows ID_{CS} and x.

Step 2: The attacker sends an authentication request message M_5^{replay} to S_j. The message contains the same information transmitted in previous communication.

Step 3: The attacker uses ID_{CS}, x and PID_i to compute
$$C_1^* = h(PID_i \parallel ID_{CS} \parallel x),$$

$r_U = D_1 \oplus h(PID_i \parallel ID_{CS} \parallel x)$,

$ID_i = D_2 \oplus h(r_U \parallel PID_i \parallel ID_{CS})$, and

$PID_i^{new} = D_3 \oplus h(ID_i \parallel x) \oplus h(r_U \parallel ID_i)$.

Step 4: After CS verifies the authenticity of ID_i and D_4, CS computes and sends

$D_{12} = h\left(PID_i^{new} \parallel ID_{cs} \parallel x\right) \oplus h\left(r_U \parallel PID_i^{new}\right)$,

$D_{13} = h\left(PID_i^{new} \parallel r_U \parallel PID_i\right) \oplus (r_S \oplus r_{SC})$, and

$D_{14} = h\left(SK_{CS} \parallel D_{12} \parallel D_{13} \parallel h(ID_i \parallel x)\right)$.

Step 5: Upon receiving the authentication response message M_8, the attacker computes

$(r_S \oplus r_{SC}) = D_{13} \oplus h\left(PID_i^{new} \parallel r_U \parallel PID_i\right)$ and

$SK_U = h(r_U \oplus r_S \oplus r_{CS})$.

As a consequence, the attacker can launch the replay attack successfully.

User Impersonation Attack

The attacker can compute a valid authentication request message which contains the correct parameters to be authenticated by CS. The description of the attack is presented below:

Step 1: The attacker knows ID_{CS}, x and PID. Thus, he/she computes

$C_1^* = h(PID_i \parallel ID_{CS} \parallel x)$.

Step 2: The attacker recovers r_U from D_1 computing

$r_U = D_1 \oplus h(PID_i \parallel ID_{CS} \parallel x)$.

Step 3: The attacker has enough information to recover sensitive data as follows

$ID_i = D_2 \oplus h(r_U \parallel PID_i \parallel ID_{CS})$ and

$PID_i^{new} = D_3 \oplus h(ID_i \parallel x) \oplus h(r_U \parallel ID_i)$.

Step 4: From this point, the attacker computes a fake authentication request message as follows:

$D_1^{fake} = C_1^* \oplus r_U^{fake}$,

$D_2^{fake} = h\left(PID_i^{new} \parallel ID_{CS} \parallel r_U^{fake}\right) \oplus ID_i$,

$D_3^{fake} = C_2^* \oplus PID_i^{fake} \oplus h\left(r_U^{fake} \parallel ID_i\right)$, and

$D_4^{fake} = h\left(ID_i \parallel PID_i^{new} \parallel PID_i^{fake} \parallel r_U^{fake} \parallel D_3^{fake}\right)$.

Step 5: The attacker sends the fake authentication request message $M_5^{fake} = \left\{PID_i^{new}, D_1^{fake}, D_2^{fake}, D_3^{fake}, D_4^{fake}\right\}$ to S_j. After S_j finalizes the process, S_j sends the authentication request message $M_6 = \left\{PID_i^{new}, D_1^{fake}, D_2^{fake}, D_3^{fake}, D_4^{fake}, PSID_j, D_5, D_6, D_7, D_8\right\}$ to CS. Upon receiving the authentication request message M_6, CS carries out the verification process of D_4^{fake} computing $h\left(ID_i \parallel PID_i^{new} \parallel PID_i^{fake} \parallel r_U^{fake} \parallel D_3^{fake}\right)$ and verifying D_4^{fake} ? $= h\left(ID_i \parallel PID_i^{new} \parallel PID_i^{fake} \parallel r_U^{fake} \parallel D_3^{fake}\right)$. It is obvious that, D_4^{fake} will pass the verification process because it contains the original ID_i and last PID_i^{new}.

2.4. Security Drawbacks

In this subsection, we expose the absence of security requirements in the scheme of Zhou et al. [30]. The security analysis was conducted based on [5,6,15]. From these references, we initialized the security analysis.

2.4.1. Fails to Provide Mutual Authentication

The scheme proposed by Zhou et al. does not provide mutual authentication. The CS verifies the identity of U_i and S_j during the third step of the authentication phase; however, neither the user nor server verifies the identity of each other. Moreover, the authentication request messages M_5 and M_6 do not contain information which establishes a relationship between U_i and S_j, as evidence to the attempt of connection.

2.4.2. Fails to Protect Secret Key

In the scheme proposed by Zhou et al., the CS uses the same secret key (x) to register users and servers. Moreover, the secret key is hidden by means of $C_1^* = h(PID_i \| ID_{CS} \| x)$ and $C_2^* = h(ID_i \| x)$; however an attacker can recover it for three reasons.

The first reason is related with the fact that CS uses the same secret key to register each user and each server, increasing the possibility of finding the correct value of x. The second reason is related with the fact that an attacker knows ID_{CS} and PID_i; this means that, each user knows two of three security parameters, decreasing the entropy to find x, in polynomial time. Finally, the low execution time of the hash function makes possible to find the secret key in polynomial time, in specific, using $C_2^* = h(ID_i \| x)$ because the attacker knows ID_i.

3. Proposed Scheme

In this section, we present our new version of a lightweight IoT-based authentication scheme in cloud computing circumstances. The scheme includes mutual authentication and key agreement to provide strong security for accessing any server of the cloud. The scheme consists of the following phases:

Registration is the process through CS creates the security parameters of each member of the system. This phase is mandatory for users and servers. The communication among participants is through a secure channel avoiding eavesdropper.

Login is the process through U_i gets access to security parameters stored in his/her SC_U. U_i needs to insert his/her SC_U, and inputs ID_i and PW_i. Then, SC_U computes and verifies the legitimacy of U_i. If the verification process is correct, U_i sends the authentication request message to S_j through an open communication channel.

Authentication is the processes by CS carries out the validation process of U_i and S_j. Moreover, CS verifies that both entities want to establish a secure communication.

Key agreement is the process by CS computes the session key for U_i and S_j. The session key is unique.

Mutual authentication is the process through U_i and S_j verifies the legitimacy of each other. In this case, U_i sends a challenge to S_j. If the response is correct, U_i knows that S_j is a member of the system.

Table 1 summarizes the notations used throughout our proposal.

Table 1. Notations of the proposed scheme.

Symbol	Description
U_i	User
S_j	Cloud server
CS	Control server
SC_i	Smart card of U_i
ID_i, SID_j, ID_{CS}	Identity of U_i, S_j, CS, respectively
PID_i, $PSID_j$	Pseudo-identity of U_i, S_j, respectively
PW_i	Password of U_i
x, y, z	Secret keys of CS. Secret keys are long integers
n_U, n_S, n_{CS}	Random nonce of U_i, S_j, CS, respectively
T_U, T_S, T_{CS}	Timestamp of U_i, S_j, CS, respectively
SK_{U-S}	Session key between U_i and S_j
$E_{SK}(\cdot)/D_{SK}(\cdot)$	Symmetric encryption/decryption using SK_{U-S}
$h(\cdot)$	Collision free one-way hash function
$\oplus$	Exclusive-OR operation
$\|$	Concatenation operation
$\Rightarrow$	Secure communication channel
$\rightarrow$	Open communication channel

3.1. Registration Phase

This phase includes user registration and server registration.

3.1.1. User Registration Sub-Phase

This sub-phase is initialized by U_i when wants to be part of the system. The details of each step are shown in Figure 2.
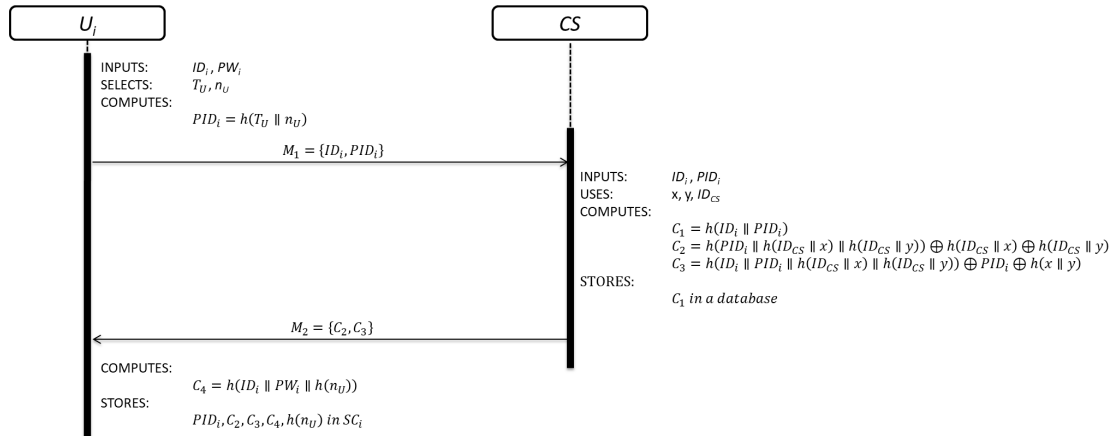
Figure 2. User registration sub-phase.

Step 1: U_i inserts his/her SC_i into a device and keys his/her ID_i and PW_i. Then, SC_i generates a random nonce (n_U), obtains the current timestamp value (T_U) and computes:

$$PID_i = h(T_U \parallel n_U) \tag{1}$$

$$U_i \Rightarrow CS : M_1 = \{ID_i, PID_i\}$$

Equation (1) is used to compute the pseudo-identity of each user.

Step 2: After receiving the registration request message M_1, CS verifies the validity of ID_i. If ID_i is valid, CS computes:

$$C_1 = h(ID_i \parallel PID_i) \tag{2}$$

$$C_2 = h(PID_i \parallel h(ID_{CS} \parallel x) \parallel h(ID_{CS} \parallel y)) \oplus h(ID_{CS} \parallel x) \oplus h(ID_{CS} \parallel y) \tag{3}$$

$$C_3 = h(ID_i \parallel PID_i \parallel h(ID_{CS} \parallel x) \parallel h(ID_{CS} \parallel y)) \oplus PID_i \oplus h(x \parallel y) \tag{4}$$

$$STORES\ C_1\ in\ a\ database$$

$$CS \Rightarrow U_i : M_2 = \{C_2,\ C_3\}$$

CS registers U_i by Equation (2). Equations (3) and (4) are security parameters for future authentication purpose.

Step 3: Upon receiving the registration response message M_2, SC_i computes:

$$C_4 = h(ID_i \parallel PW_i \parallel h(n_U)) \tag{5}$$

$$STORES\ PID_i, C_2, C_3, C_4, h(n_U)\ in\ SC_i$$

SC_i computes the local user authentication parameter using Equation (5) and stores all the security parameters received from CS. The user registration sub-phase is over.

3.1.2. Cloud Server Registration

This sub-phase is initialized by each S_j in order to be part of the system. The details of each step are shown in Figure 3.
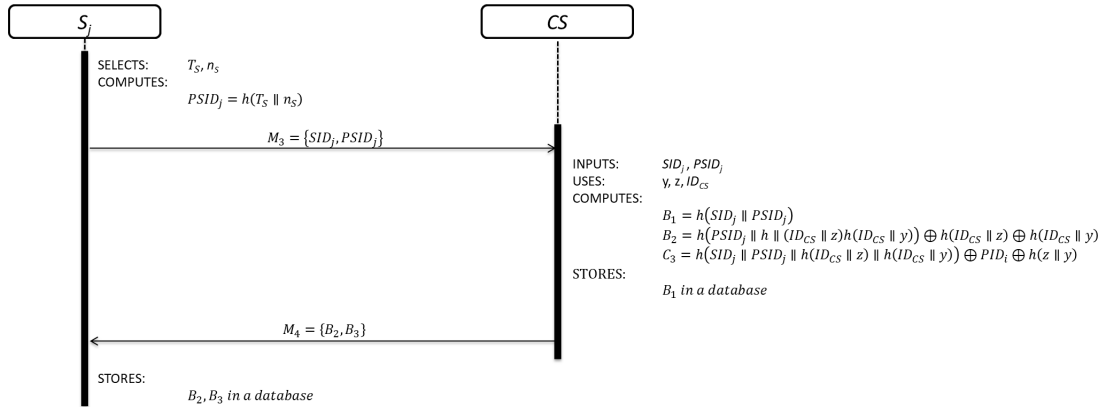

Figure 3. Server registration sub-phase.

Step 1: S_j generates a random nonce n_S, obtains the current timestamp value T_S and computes:

$$PSID_j = h(T_S \parallel n_S) \tag{6}$$

$$S_j \Rightarrow CS : M_3 = \left\{ SID_j, PSID_j \right\}$$

Equation (6) is used to compute the pseudo-identity of each server.

Step 2: After receiving the registration request message M_3, CS verifies the validity of SID_j. If SID_j is correct, CS computes:

$$B_1 = h\left(SID_j \parallel PSID_j \right) \tag{7}$$

$$B_2 = h\left(PSID_j \parallel h(ID_{CS} \parallel z) \parallel h(ID_{CS} \parallel y) \right) \oplus h(ID_{CS} \parallel z) \oplus h(ID_{CS} \parallel y) \tag{8}$$

$$B_3 = h\left(SID_j \parallel PSID_j \parallel h(ID_{CS} \parallel z) \parallel h(ID_{CS} \parallel y) \right) \oplus PSID_j \oplus h(z \parallel y) \tag{9}$$

$$STORES\ B_1\ in\ a\ database$$

$$CS \Rightarrow S_j : M_4 = \{B_2, B_3\}$$

CS registers S_j by Equation (7). Equations (8) and (9) are security parameters for future authentication purpose.

Step 3: After receiving the registration response message M_4, S_j stores B_2 and B_3.

3.2. Login Phase

Once U_i was registered, U_i can connect to any server of the cloud by initiating the login phase. The details are shown in Figure 4.

Step 1: In order to start the authentication phase, U_i must first complete the login phase. Firstly, U_i inserts SC_i and keys his/her ID_i^* and PW_i^*.

Step 2: Then, SC_i computes and checks:

$$C_4^* = h\left(ID_i^* \parallel PW_i^* \parallel h(n_U) \right)\ ? = C_4 \tag{10}$$

Equation (10) is used to verify the legitimacy of U_i by SC_i for getting access to security parameters provided by S_j.

Step 3: If the verification process is correct, SC_i generates a random nonce n_U^{new}, obtains the current timestamp value T_U^{new} and computes:

$$D_1 = C_2 \oplus ID_i \tag{11}$$

$$D_2 = C_3 \oplus h\left(T_U^{new} \parallel ID_i\right) \oplus h(n_u^{new}) \tag{12}$$

$$U_i \rightarrow S_j : M_5 = \left\{T_U^{new}, D_1, PID_i, D_2\right\}$$

Equations (11) and (12) contain U_i's information which will be used by CS to verify its legitimacy. Finally, U_i sends the user authentication request message M_5 to S_j through an open communication channel.

Step 4: After receiving the user authentication request message M_5, S_j generates a random nonce n_S^{new}, obtains the current timestamp value T_S^{new} and computes:

$$D_3 = B_2 \oplus SID_j \tag{13}$$

$$D_4 = B_3 \oplus h\left(T_S^{new} \parallel SID_j\right) \oplus h\left(n_S^{new}\right) \tag{14}$$

$$D_5 = h\left(PID_i \parallel T_U^{new} \parallel SID_j \parallel PSID_j \parallel T_S^{new}\right) \tag{15}$$

$$S_j \rightarrow CS : M_6 = \left\{T_U^{new}, D_1, PID_i, D_2, T_S^{new}, D_3, PSID_j, D_4, D_5\right\}$$

Equations (13) and (14) contain S_j's information which will be used by CS to verify its legitimacy. Equation (15) contains information about U_i and S_j as evidence of its connection attempt. Finally, S_j sends the authentication request message M_6 to CS through an open communication channel.
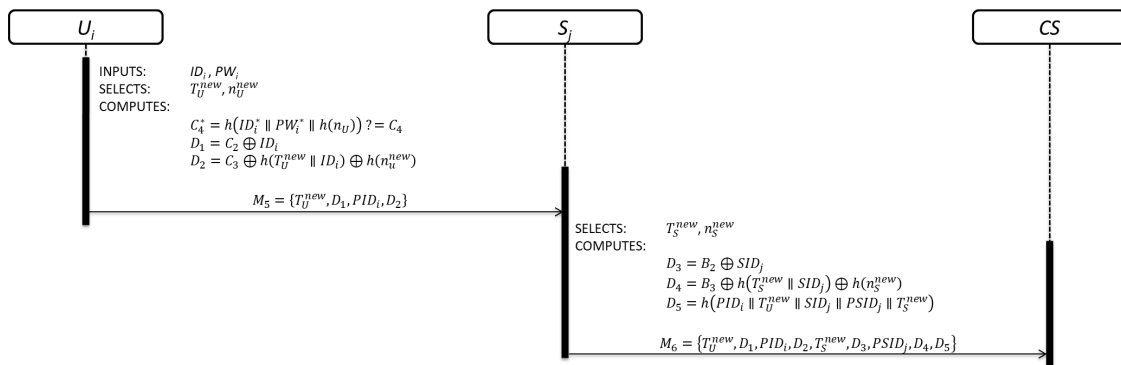

Figure 4. Login phase.

3.3. Authentication Phase

This phase is divided in three sub-phases. The details of user authentication, server authentication and evidence of connection attempt are shown in Figure 5.

3.3.1. User Authentication

Step 1: Upon receiving the authentication request message M_6, CS checks the freshness of the message by means of T_U^{new}. If the verification process is positive, CS computes:

$$C_2^* = h\left(PID_i^* \parallel h(ID_{CS} \parallel x) \parallel h(ID_{CS} \parallel y)\right)^* \oplus h(ID_{CS} \parallel x) \oplus h(ID_{CS} \parallel y)$$

where PID_i^* is the pseudo-identity of U_i contained in message M_2. CS computed C_2^* using PID_i^* and Equation (3).

Step 2: *CS* verifies the legitimacy of U_i by means of C_1 as follows:

$$D_1 = C_2^* \oplus ID_i$$
$$D_1 = h\left(PID_i^* \,\|\, h(ID_{CS} \,\|\, x) \,\|\, h(ID_{CS} \,\|\, y)\right)^* \oplus h(ID_{CS} \,\|\, x) \oplus h(ID_{CS} \,\|\, y) \oplus ID_i \tag{16}$$
$$ID_i^* = h\left(PID_i^* \,\|\, h(ID_{CS} \,\|\, x) \,\|\, h(ID_{CS} \,\|\, y)\right)^* \oplus h(ID_{CS} \,\|\, x) \oplus h(ID_{CS} \,\|\, y) \oplus D_1$$

$$C_1^* = h\left(ID_i^* \,\|\, PID_i\right) ? = C_1 \tag{17}$$

Equation (16) is used to recover ID_i^* from D_1. Equation (17) is used to verify the legitimacy of U_i using ID_i^* and C_1. If the verification process is correct, *CS* continues with the next step; otherwise, *CS* finalizes the process.

Step 3: After verifying the legitimacy of U_i, *CS* recovers $h\left(n_U^{new}\right)$ as follows:

$$h\left(n_U^{new}\right)^* = h\left(ID_i^* \,\|\, PID_i^* \,\|\, h(ID_{CS} \,\|\, x) \,\|\, h(ID_{CS} \,\|\, y)\right)^* \oplus PID_i^* \oplus h(x \,\|\, y) \oplus h\left(T_U^{new} \,\|\, ID_i^*\right)^* \oplus D_2 \tag{18}$$

Equation (18) is used to recover $h\left(n_U^{new}\right)^*$ from D_2.

Step 4: *CS* computes C_1^{new} with T_U^{new} and $h\left(n_U^{new}\right)^*$ using Equation (19). Then, *CS* updates C_1 in the database:

$$C_1^{new} = h\left(ID_i \,\|\, h\left(T_U^{new} \,\|\, C_1 \,\|\, h\left(n_U^{new}\right)\right)\right) \tag{19}$$

3.3.2. Server Authentication

Step 1: Upon finalizing the user authentication process, *CS* checks the freshness of the message by means of T_S^{new}.

Step 2: If the verification process is positive, *CS* verifies the legitimacy of S_j as follows:

$$D_3 = B_2 \oplus SID_j$$
$$B_2 = h\left(PSID_j^* \,\|\, h(ID_{CS} \,\|\, z) \,\|\, h(ID_{CS} \,\|\, y)\right)^* \oplus h(ID_{CS} \,\|\, z) \oplus h(ID_{CS} \,\|\, y) \oplus SID_j \tag{20}$$
$$SID_j^* = h\left(PSID_j^* \,\|\, h(ID_{CS} \,\|\, z) \,\|\, h(ID_{CS} \,\|\, y)\right)^* \oplus h(ID_{CS} \,\|\, z) \oplus h(ID_{CS} \,\|\, y) \oplus D_3$$

$$B_1^* = h\left(SID_j^* \,\|\, PSID_j^*\right) ? = B_1 \tag{21}$$

Equation (20) is used to recover SID_j^* from D_3, using $PSID_j^*$. Then, *CS* computes Equation (21) to obtain B_1^* and compares it with B_1. If the verification process is correct, *CS* continues with the process, otherwise, *CS* finalizes the process.

Step 3: After verifying the legitimacy of S_j, *CS* recover $h\left(n_S^{new}\right)$ as follows:

$$h\left(n_S^{new}\right)^* = h\left(SID_j^* \,\|\, PSID_j^* \,\|\, h(ID_{CS} \,\|\, z) \,\|\, h(ID_{CS} \,\|\, y)\right)^* \oplus PSID_j^* \oplus h(z \,\|\, y) \oplus h\left(T_S^{new} \,\|\, SID_j^*\right) \oplus D_4 \tag{22}$$

Finally, Equation (22) is used to recover $h\left(n_S^{new}\right)^*$ from D_4.

3.3.3. Evidence of Connection Attempt

Step 1: *CS* corroborates that U_i wants to establish a connection with S_j as follows:

$$D_5^* = h\left(PID_i^* \,\|\, T_U^{new} \,\|\, SID_j^* \,\|\, PSID_j^* \,\|\, T_S^{new}\right)^* ? = D_5 \tag{23}$$

in this case, *CS* has evidence of the connection attempt between U_i and S_j. It is important to note that, Equation (23) requires the fresh timestamp from U_i and S_j. Moreover, D_5 contains PID_i, SID_j and $PSID_j$ which demonstrate the interest of the two entities for establishing a secure communication.
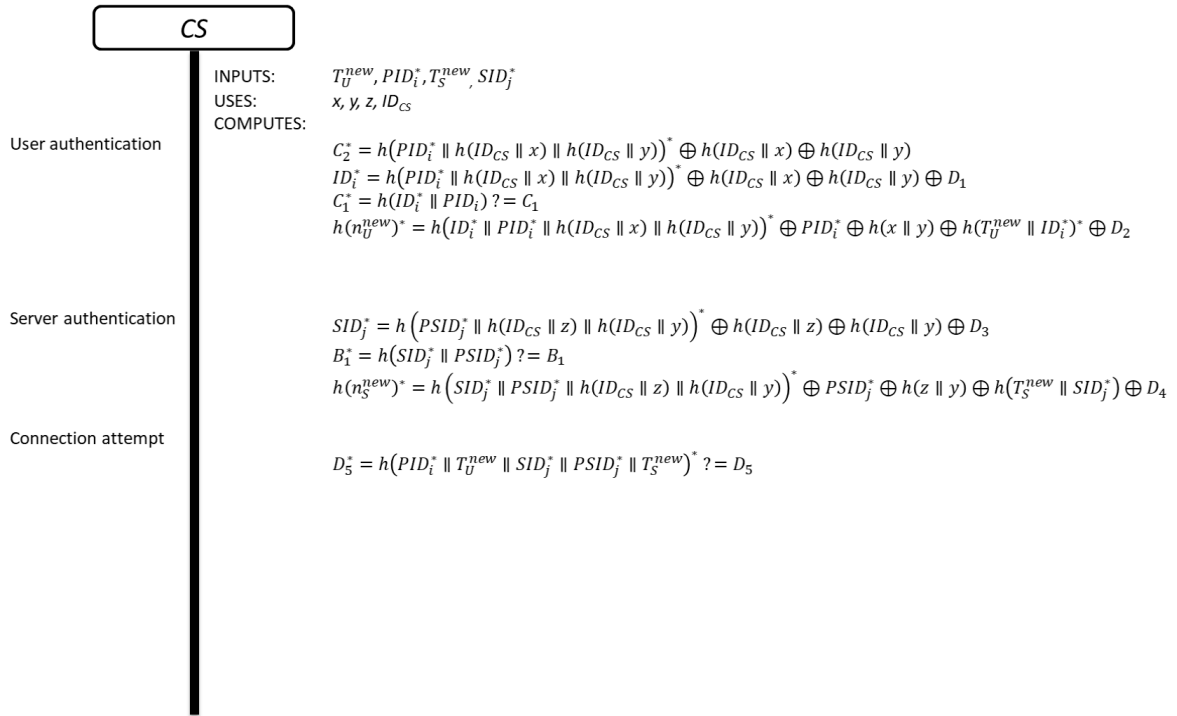
Figure 5. User authentication, server authentication and evidence of connection attempt sub-phases.

3.4. Key Agreement Phase

This phase is divided in three sub-phases. The details of each phase are shown in Figure 6.
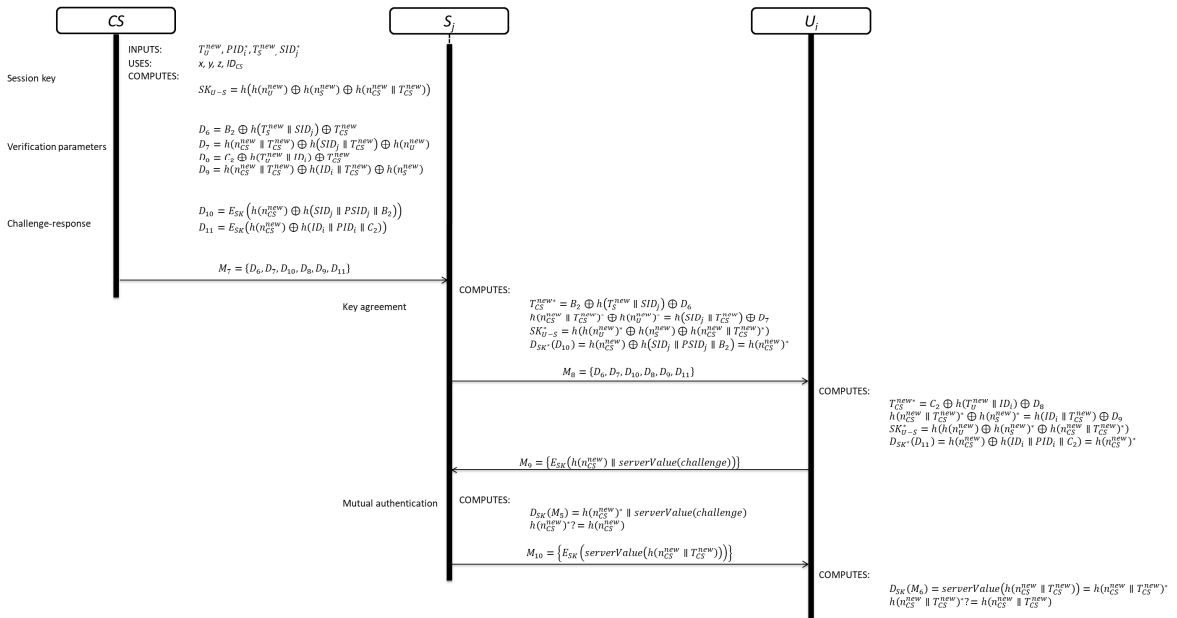
Figure 6. Key agreement and mutual authentication phases.

3.4.1. Session Key Creation

In this sub-phase, CS computes the session key between U_i and S_j as follows:

Step 1: CS generates a random nonce n_{CS}^{new} and computes the session key SK_{U-S} as follows:

$$SK_{U-S} = h\left(h\left(n_U^{new}\right) \oplus h\left(n_S^{new}\right) \oplus h\left(n_{CS}^{new} \parallel T_{CS}^{new}\right)\right) \tag{24}$$

Equation (24) is used to compute the session key. The session key contains security parameters generated by U_i, S_j and CS which represents the relationship among all the participants.

Step 2: CS computes the verification parameters for S_j and U_i as follows:

$$D_6 = B_2 \oplus h\left(T_S^{new} \parallel SID_j\right) \oplus T_{CS}^{new} \tag{25}$$

$$D_7 = h\left(n_{CS}^{new} \parallel T_{CS}^{new}\right) \oplus h\left(SID_j \parallel T_{CS}^{new}\right) \oplus h\left(n_U^{new}\right) \tag{26}$$

$$D_8 = C_2 \oplus h\left(T_U^{new} \parallel ID_i\right) \oplus T_{CS}^{new} \tag{27}$$

$$D_9 = h\left(n_{CS}^{new} \parallel T_{CS}^{new}\right) \oplus h\left(ID_i \parallel T_{CS}^{new}\right) \oplus h\left(n_S^{new}\right) \tag{28}$$

where D_6 and D_7 are for S_j, while D_8 and D_9 are for U_i. Equations (25) to (28) contain information generated by CS for computing the session key.

Step 3: CS computes the challenge-response message for S_j and U_i as follows:

$$D_{10} = E_{SK}\left(h\left(n_{CS}^{new}\right) \oplus h\left(SID_j \parallel PSID_j \parallel B_2\right)\right) \tag{29}$$

$$D_{11} = E_{SK}\left(h\left(n_{CS}^{new}\right) \oplus h(ID_i \parallel PID_i \parallel C_2)\right) \tag{30}$$

$$CS \rightarrow S_j : M_7 = \{D_6, D_7, D_{10}, D_8, D_9, D_{11}\}$$

CS computed the challenge-response message for each entity using the session key; this means that, a legitimate participant can recover the security parameters to construct the session key. Finally, CS sends the authentication response message M_7 to S_j through an open communication channel.

3.4.2. Server Session Key

Step 1: After receiving M_7, S_j computes and verifies the freshness of M_7:

$$T_{CS}^{new*} = B_2 \oplus h\left(T_S^{new} \parallel SID_j\right) \oplus D_6 \tag{31}$$

Equation (31) is used to extract T_{CS}^{new*} from D_6.

Step 2: If M_7 is fresh, S_j computes:

$$h\left(n_{CS}^{new} \parallel T_{CS}^{new}\right)^* \oplus h\left(n_U^{new}\right)^* = h\left(SID_j \parallel T_{CS}^{new}\right) \oplus D_7 \tag{32}$$

$$SK_{U-S}^* = h\left(h\left(n_U^{new}\right)^* \oplus h\left(n_S^{new}\right) \oplus h\left(n_{CS}^{new} \parallel T_{CS}^{new}\right)^*\right)$$

$$h\left(n_{CS}^{new}\right)^* = h\left(n_{CS}^{new}\right) \oplus h\left(SID_j \parallel PSID_j \parallel B_2\right) = D_{SK^*}(D_{10}) \tag{33}$$

$$S_j \rightarrow U_i : M_8 = \{D_8, D_9, D_{11}\}$$

at this point, S_j knows the session key $\left(SK_{U-S}^*\right)$ and the value $h\left(n_{CS}^{new}\right)^*$. Finally, S_j sends M_8 to U_i through an open communication channel.

3.4.3. User Session Key

Step 1: Upon receiving the user authentication response message M_8, U_i computes and verifies the freshness of M_8:

$$T_{CS}^{new*} = C_2 \oplus h\left(T_U^{new} \parallel ID_i\right) \oplus D_8 \tag{34}$$

Step 2: If M_8 is fresh, U_i computes:

$$h\left(n_{CS}^{new} \parallel T_{CS}^{new}\right)^* \oplus h\left(n_S^{new}\right)^* = h\left(ID_i \parallel T_{CS}^{new}\right) \oplus D_9 \tag{35}$$

$$SK^*_{U-S} = h\left(h\left(n^{new}_U\right) \oplus h\left(n^{new}_S\right)^* \oplus h\left(n^{new}_{CS} \| T^{new}_{CS}\right)^*\right)$$

$$D_{SK^*}(D_{11}) = h\left(n^{new}_{CS}\right) \oplus h(ID_i \| PID_i \| C_2) = h\left(n^{new}_{CS}\right)^* \tag{36}$$

U_i knows the session key and the value $h\left(n^{new}_{CS}\right)^*$.

3.5. Mutual Authentication

Step 1: U_i sends the challenge message M_9 to S_j. M_9 contains $h\left(n^{new}_{CS}\right)$ as proof of his/her legitimacy and requests the response:

$$U_i \to S_j : M_9 = \left\{E_{SK}\left(h\left(n^{new}_{CS}\right) \| serverValue(challenge)\right)\right\}$$

Step 2: Upon receiving the challenge message M_9, S_j computes

$$h\left(n^{new}_{CS}\right)^* \| serverValue(challenge) = D_{SK}(M_9) \tag{37}$$

$$h\left(n^{new}_{CS}\right)^* ? = h\left(n^{new}_{CS}\right) \tag{38}$$

$$S_j \to U_i : M_{10} = \left\{E_{SK}\left(serverValue\left(h\left(n^{new}_{CS} \| T^{new}_{CS}\right) \oplus h\left(n^{new}_U\right)\right)\right)\right\}$$

S_j knows that U_i is the user who requested the user authentication by means of Equations (37) and (38). Then, S_j sends the response to U_i.

Step 3: Upon receiving the message M_{10}, U_i computes and verifies the legitimacy of S_j as follows:

$$h\left(n^{new}_{CS} \| T^{new}_{CS}\right)^* \oplus h\left(n^{new}_U\right) = serverValue\left(h\left(n^{new}_{CS} \| T^{new}_{CS}\right) \oplus h\left(n^{new}_U\right)\right) = D_{SK}(M_{10}) \tag{39}$$

$$h\left(n^{new}_{CS} \| T^{new}_{CS}\right)^* ? = h\left(n^{new}_{CS} \| T^{new}_{CS}\right) \tag{40}$$

finally, U_i recovers the response from M_{10} using Equation (39) and verifies the legitimacy of S_j by means of Equation (40).

Step 4: U_i replaces PID_i, $h(n_U)$, and $C_4 = h(ID_i \| PW_i \| h(n_U))$ with $PID^{new}_i = \left(h\left(T^{new}_U \| h(ID_i \| PID_i) \| n^{new}_U\right)\right)$, $h\left(n^{new}_U\right)$, and $C^{new}_4 = h\left(ID_i \| PW_i \| h\left(n^{new}_U\right)\right)$, respectively.

3.6. Password Change Phase

When U_i wants to change or to update his/her password, he/she needs to key his/her ID_i and PW_i. Then, SC_i computes Equation (10) to verify his/her legitimacy. If the verification process is correct, U_i keys PW^{new}_i and SC_i computes Equation (40):

$$C^{new}_4 = h\left(ID_i \| PW^{new}_i \| h(n_u)\right) \tag{41}$$

UPDATES C_4

4. Security Analysis and Performance Evaluation

In this section, we carry out the security analysis and performance evaluation comparison of our proposal. The security analysis includes an informal cryptanalysis, security of session key and countermeasures to improve security. The performance evaluation includes computational- and communication-cost comparison with Zhou et al.'s, Amin et al.'s and Xue et al.'s schemes.

4.1. Informal Cryptanalysis

In this sub-section, we analyse the security of our proposal using informal security analysis.

4.1.1. User Anonymity

In our scheme, U_i sends $PID_i = h(T_U \parallel n_U)$ to S_j instead of ID_i in clear text. Moreover, PID_i is updated after finalizing the user authentication sub-phase with T_U^{new} and n_U^{new}, keeping the identity of each user anonym. In consequence, when the user sends the authentication request message to S_j, the PID_i will be different. Furthermore, an attacker cannot recover ID_i from PID_i, D_1, or D_2 without security parameters. Therefore, the scheme provides user anonymity.

4.1.2. Off-line User Identity and Password Guessing Attack

In the case that a malicious user obtains SC_i, he/she can recover $PID_i, C_2, C_3, C_4, h(n_U)$ [33,34]. However, he/she cannot obtain sensitive data from C_2 because nobody knows x, y, z, and ID_{CS}. From C_3 the attacker can extract PID_i which it is stored in SC_i. In this case, he/she is not capable to extract sensitive data from SC_i.

4.1.3. Privileged Insider Attack

In our scheme, security parameters are personalized using data from each user or cloud server, making more complex the possibility to know secret keys of CS. If a malicious user tries to extract x and y from C_2 or C_3, he/she needs to recover $h(ID_{CS} \parallel x)$ or $h(ID_{CS} \parallel y)$ from Equations (3) or (4):

$$C_2 = h(PID_i \parallel h(ID_{CS} \parallel x) \parallel h(ID_{CS} \parallel y)) \oplus h(ID_{CS} \parallel x) \oplus h(ID_{CS} \parallel y)$$

$$C_3 = h(ID_i \parallel PID_i \parallel h(ID_{CS} \parallel x) \parallel h(ID_{CS} \parallel y)) \oplus PID_i \oplus h(x \parallel y)$$

but the attacker only know PID_i. Moreover, CS does not include its ID_{CS} in any messages or shares it in clear text. Thus, the scheme resists this attack.

4.1.4. Impersonation Attack

In the case that an attacker obtains $PID_i, C_2, C_3, C_4, h(n_U)$ from SC_i [33,34] and $M_5 = \{T_U^{new}, D_1, PID_i, D_2\}$ the attacker cannot create a valid authentication request message by any type of combination of the security parameters. In this case, the malicious user computes $D_1^{fake} = C_2 \oplus ID_i^{fake}$ using ID_i^{fake} but he/she cannot compute a valid PID_i and C_3 which are required to compute a valid D_2. In consequence, the attacker cannot impersonate a legal user.

4.1.5. Replay Attack

In this attack, the malicious user needs to know previous authentication request message $M_5 = \{T_U^{new}, D_1, PID_i, D_2\}$. However, our scheme uses random nonce (n_U) and timestamp (T_U) to avoid replay attack. The control server verifies the freshness of the timestamp every time.

4.2. Security of Session Key

A key purpose of an authentication scheme is the establishment of a session key, so the session key should be protected against known-key security and forward secrecy [22].

4.2.1. Known-key Security

In our scheme, CS computes new session key every time the authentication is correct. This means that, CS uses the new random nonce $\left(n_U^{new}\right)$, $\left(n_S^{new}\right)$ and $\left(n_{CS}^{new}\right)$, and its current timestamp $\left(T_{CS}^{new}\right)$ to compute a fresh session key, avoiding the compromise of previous session keys. If the attacker knows $M_7 = \{D_6, D_7, D_{10}, D_8, D_9, D_{11}\}$, he/she cannot compute a valid session key (SK_{U-S}) without random nonce and $CS's$ timestamp. Even though the attacker knows past session key $\left(SK_{U-S}^{old}\right)$, he/she cannot compute the new session key by means of any type of combination.

4.2.2. Forward Secrecy

In our scheme, CS computes the session key without the use of secret keys (x, y, z), avoiding compromise its security in case that an attacker knows the secret keys. Let us suppose that, an attacker knows (x, y, z), he/she cannot compute the correct session key because it does not contain the secret keys. Thus, the attacker cannot create a valid session key.

4.3. Countermeasures

4.3.1. Local Protection against Malicious Users

In our scheme, SC_i verifies the legitimacy of U_i by means of Equation (10). This mean that U_i must be authenticated by SC_i before it computes the user authentication request message [22].

4.3.2. Mutual Authentication

In our scheme, U_i and S_j verify that each other is a legitimate user in the system and want to establish a secure communication through Equations (37) to (40). In this case, U_i sends a challenge to S_j for carrying out the mutual authentication process. The response message contains $\left(n_{CS}^{new} \parallel T_{CS}^{new} \right)$ which represents the fresh of the communication. Moreover, U_i knows the same value, thus avoiding a man-in-the-middle attack.

4.3.3. Evidence of Connection Attempt

In our scheme, S_j computes D_5, using Equation (15), which contains information from U_i and S_j, making unique the value of D_5. Then, CS verifies the connection attempt between U_i and S_j by means of Equation (23). Moreover, CS computes the session key using information of U_i, S_j and CS.

4.4. Security Comparison

This sub-section presents the security comparison of the proposed scheme with Zhou et al.'s scheme, Amin et al.'s scheme and Xue et al.'s schemes in terms of security properties. Table 2 lists comparative results.

Table 2. Security comparison.

Security Property	Xue et al.	Amin et al.	Zhou et al.	Our Scheme
Provide evidence of connection attempt	fails	fails	fails	success
Provide mutual authentication	fails	fails	fails	success
Provide user anonymity	fails			success
Resist impersonation attack	fails	fails	fails	success
Resist off-line user identity/password attack	fails			success
Resist privileged-insider attack	fails	fails		success
Resist replay attack			fails	success

According to Table 2, it is clear that previous works are vulnerable to different attacks and fails to provide mutual authentication between the server and the user. Moreover, previous works do not provide evidence of connection attempts. In consequence, our protocol resists very well-known attacks, provides evidences of connection attempts, mutual authentication and user anonymity.

4.5. Computational-cost Comparison

This sub-section presents the performance evaluation of the proposed scheme with Zhou et al.'s scheme, Amin et al.'s scheme and Xue et al.'s scheme in terms of execution-time. The evaluation of each scheme was based on the following considerations:

- T_h represents a hash function.
- T_S represents an encryption/decryption operation using AES algorithm.
- The execution time for T_h is case 1: 0.00517 ms [30] and case 2: 0.0000328 ms [32].
- The execution time for T_S is case1: 0.02148 ms [30] and case2: 0.0214385 ms [32].

Table 3 summarizes the operations carried out by U_i, S_j and CS during the registration, login and authentication phases. The execution time required by U_i, S_j and CS during each phase is shown in Table 4.

Table 3. Performance comparison.

Phase		Xue et al.	Amin et al.	Zhou et al.	Our Scheme
Registration	U_i	$3T_h$	$3T_h$	$3T_h$	$2T_h$
	S_j	$0T_h$	$0T_h$	$0T_h$	$1T_h$
	CS	$4T_h$	$4T_h$	$4T_h$	$12T_h$
Login	U_i	$6T_h$	$6T_h$	$6T_h$	$3T_h$
	S_j	$3T_h$	$1T_h$	$3T_h$	$3T_h$
	CS	$0T_h$	$0T_h$	$0T_h$	$0T_h$
Authentication	U_i	$3T_h$	$3T_h$	$4T_h$	$4T_h + 3T_s$
	S_j	$3T_h$	$3T_h$	$4T_h$	$2T_h + 3T_s$
	CS	$14T_h$	$10T_h$	$19T_h$	$21T_h + 2T_s$
Total		$36T_h$	$30T_h$	$43T_h$	$48T_h + 8T_s$

Table 4. Execution-time by participant.

		Xue et al.			Amin et al.			Zhou et al.			Our		
		Op	Case 1	Case 2	Op	Case 1	Case 2	Op	Case 1	Case 2	Op	Case 1	Case 2
R	U_i	3Th	0.01551	0.0000984	3Th	0.01551	0.0000984	3TH	0.01551	0.0000984	2Th	0.01034	0.0000656
	S_j	0Th	0	0	0Th	0	0	0TH	0	0	1Th	0.00517	0.0000328
	CS	4Th	0.02068	0.0001312	4Th	0.02068	0.0001312	4TH	0.02068	0.0001312	12Th	0.06204	0.0003936
L	U_i	6Th	0.03102	0.0001968	6Th	0.03102	0.0001968	6Th	0.03102	0.0001968	3Th	0.01551	0.0000984
	S_j	3Th	0.01551	0.0000984	1Th	0.00517	0.0000328	3Th	0.01551	0.0000984	3Th	0.01551	0.0000984
	CS	0Th	0	0	0Th	0	0	0Th	0	0	0Th	0	0
A	U_i	3Th	0.01551	0.0000984	3Th	0.01551	0.0000984	4Th	0.02068	0.0001312	4Th + 3Ts	0.08512	0.0644467
	S_j	3Th	0.01551	0.0000984	3Th	0.01551	0.0000984	4Th	0.02068	0.0001312	2Th + 3Ts	0.07478	0.0643811
	CS	14Th	0.07238	0.0004592	10Th	0.0517	0.000328	19Th	0.09823	0.0006232	21Th + 2Ts	0.15153	0.0435658
Total		36Th	0.18612	0.0011808	30Th	0.1551	0.000984	43Th	0.22231	0.0014104	48Th + 8Ts	0.420000	0.173082

*R = Registration phase, L = Login phase, A = Authentication phase, Op = Number of operations.

From Table 3, it is easy to see that our scheme requires more computational operations than previous works. However, it is necessary to compute the execution-time in a real scenario. For that reason, we used the execution-time described in [30] and [32] to compare the performance of each scheme. The execution-time results are shown in Table 4, and it is clear that the execution time depends on: (a) the characteristics of each device, (b) the cryptography libraries and (c) the computational load. In our scheme, CS is the participant which computes more computational operations because we assume that has more resources than U_i and S_j. Moreover, U_i computes more computational operations than S_j because we assume that U_i will request few connections per day; however, S_j will receive many request for connection, considering a high volume of users it is necessary that S_j computes few computational operations. According with the results summarized in Table 4, the scheme with les computational operations was proposed by Amin et al. [1].

In fact, the execution time of each device must be considered as show in Table 4. The computational operations evaluated using case 2 gives better results than case 1. Under this situation, our scheme requires 40% less time which represents an acceptable performance, less than 0.1862 ms [30].

Figure 7 shows the computational-cost comparison by participant and scheme. In this case, we used the execution-time of the case 1. The main difference between our scheme and previous works is the inclusion of symmetric operations during the authentication phase. The symmetric operations are used to provide mutual authentication between U_i and S_j, increasing the security of the proposed scheme.
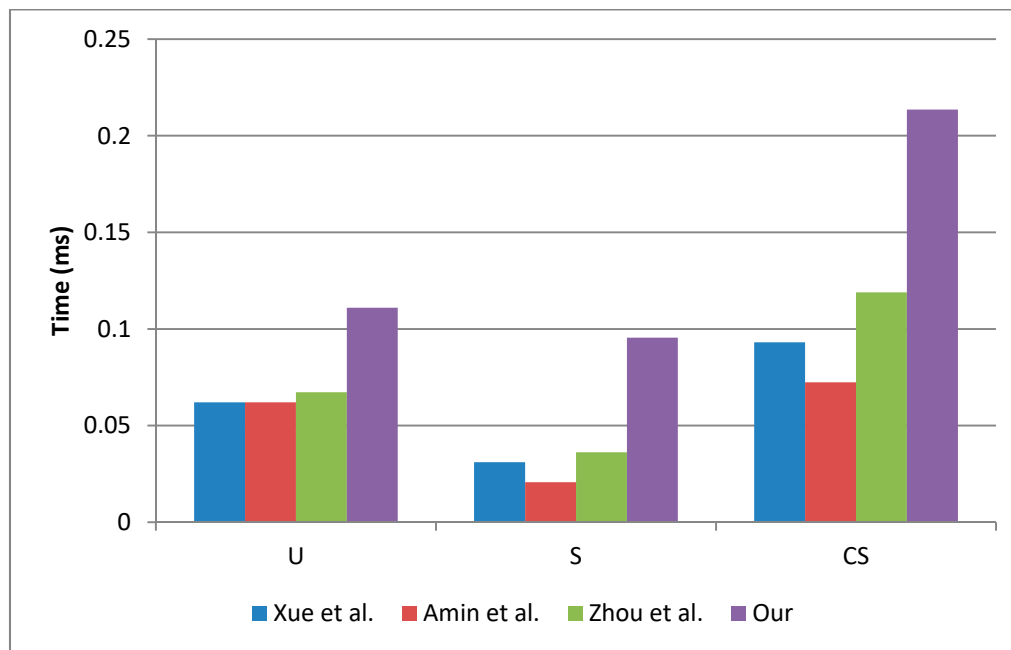
Figure 7. Computational-cost comparison by participant.

4.6. Communication-Cost Comparison

In this sub-section, we compare the communication-cost of our scheme with Zhou et al.'s scheme, Amin et al.'s scheme and Xue et al.'s scheme in terms of message length. For conventional comparison, we assume two bit length cases:

- Case 1: any identity, password, pseudo-identity, timestamp, random nonce, and hash output are 128 bits.
- Case 2: any identity, password, pseudo-identity, timestamp, random nonce, and hash output are 256 bits.
- The block length of the symmetric encryption is 128 bits.

Table 5 summarizes the message length by each entity during the scheme.

Table 5. Communication cost comparison.

		Xue et al.			Amin et al.			Zhou et al.			Our Scheme		
		Length	Case 1	Case 2	Length	Case 1	Case 2	Length	Case 1	Case 2	Length	Case 1	Case 2
R	U_i	3	384	768	2	256	512	2	256	512	2	256	512
	S_j	2	256	512	2	256	512	2	256	512	2	256	512
	CS	2	256	512	3	384	768	6	768	1536	4	512	1024
	ST		896	1792		896	1792		1280	2560		1024	2048
L	U_i	6	768	1536	5	640	1280	5	640	1280	4	512	1024
	S_j	11	1408	2816	9	1152	2304	10	1280	2560	9	1152	2304
	CS	0	0	0	0	0	0	0	0	0	0	0	0
	ST		2176	4352		1792	3584		1920	3840		1664	3328
A	U_i	0	0	0	0	0	0	0	0	0	2	256	512
	S_j	2	256	512	2	256	512	3	384	768	5	640	1280
	CS	4	512	1024	4	512	1024	6	768	1536	6	768	1536
	ST		768	1536		768	1536		1152	2304		1664	3328
	T	30	3840	7680	27	3456	6912	34	4352	8704	34	4352	8704

*R = Registration phase, L = Login phase, A = Authentication phase, ST = Subtotal, T = Total.

From Table 5, we see very clearly that our scheme requires the same message length as Zhou et al.'s scheme, 4352 bits or 8704 bits. However, our scheme provides mutual authentication and evidence of connection attempt, which requires more information to share among participants. If we pay attention phase by phase, our scheme requires less message length during the registration phase than Zhou et al.'s scheme, making our proposal more efficient. In fact, our proposal requires less message length in the login phase than previous works, making it more efficient. After achieving the performance evaluation of the proposed scheme, it is possible to confirm that the proposal has good performance.

5. Conclusions

In this paper, we demonstrated that Zhou et al.'s scheme is not secure against insider, replay and user impersonation attacks. Moreover, we found security drawbacks which make the scheme proposed by Zhou et al. insecure for IoT in cloud computing circumstances. As a consequence, we propose a new scheme to remedy the security vulnerabilities and drawbacks of Zhou et al.'s scheme.

The new scheme achieves mutual authentication and key agreement, providing secure access to cloud servers. Moreover, the proposal keeps the user identity anonymous against eavesdroppers, provides security for the session key and includes a challenge-response method. In addition, the new scheme includes a sub-phase called evidence connection attempt which proves to the control server any connection attempt between a user and a server.

Furthermore, our performance evaluation demonstrates that our scheme does not require high computational power or several messages to achieve security requirements. On the contrary, the proposed scheme requires less communication-cost than Zhou et al.'s, Amin et al.'s and Xue et al.'s schemes in the registration and login phases, and the computational-cost is acceptable considering the security characteristics included in the scheme. Thus, the scheme meets the security requirements for a secure IoT-based authentication scheme in cloud computing circumstance, enhancing security of previous works.

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

References

1.　Amin, R.; Kumar, N.; Biswas, G.P.; Iqbal, R.; Chang, V. A light weight authentication protocol for IoT-enabled devices in distributed cloud computing environment. *Future Gener. Comput. Syst.* **2018**, *78*, 1005–1019. [CrossRef]

2.　Noura, M.; Atiquzzman, M.; Gaedke, M. Interoperability in internet of things: Taxonomies and open challenges. *Mob. Netw. Appl.* **2018**. [CrossRef]

3.　Foster, I.; Zhao, Y.; Raicu, I.; Lu, S. Cloud computing and grid computing 360-degree compared. In Proceedings of the Workshop on Grid Computing Environments (GCE), Austin, TX, USA, 12–16 November 2008. [CrossRef]

4.　Sova, P. Cloud computing in brief. *IOSR J. Comput. Eng.* **2016**, *18*, 101–103.

5.　Ferrag, M.A.; Maglaras, L.A.; Janicke, H.; Jiang, J.; Shu, L. Authentication protocols for internet of things: A comprehensive survey. *Secur. Commun. Netw.* **2017**. [CrossRef]

6.　El-Hajj, M.; Fadlallah, A.; Chamoun, M.; Serhrouchni, A. A survey of internet of things (IoT) Authentication schemes. *Sensors* **2019**, *19*, 1141. [CrossRef]

7.　Yu, W.; Liang, F.; He, X.; Grant Hatcher, W.; Lu, C.; Lin, J.; Yang, X. A survey on the edge computing for the internet of things. *IEEE Access* **2017**, *6*, 6900–6919. [CrossRef]

8.　Fernández Maimó, L.; Huertas Celdrán, A.; Perales Gómez, A.L.; García Clemente, F.J.; Weimer, J.; Lee, I. Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments. *Sensors* **2019**, *19*, 1114. [CrossRef]

9.   Baker, S.B.; Xiang, W.; Atkinson, I. Internet of things for smart healthcare: Technologies, challenges, and opportunities. *IEEE Access* **2017**, *5*, 26521–26544. [CrossRef]

10.   Jang, Q.; Ma, J.; Yang, C.; Ma, X.; Shen, J.; Chaudhry, S.A. Efficient end-to-end authentication protocol for wearable health monitoring systems. *Comput. Electr. Eng.* **2017**, *63*, 182–195. [CrossRef]

11.   Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of things for smart cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]

12.   Perera, C.; Harold Liu, C.; Jayawardena, S.; Chen, M. A survey on internet of things from industrial market perspective. *IEEE Access* **2015**, *2*, 1660–1679. [CrossRef]

13.   El-Sayed, H.; Sankar, S.; Prasad, M.; Puthal, D.; Gupta, A.; Mohanty, M.; Lin, C.T. Edge of things: The big picture on the integration of edge, IoT and the cloud in a distributed computing environment. *IEEE Access* **2017**, *6*, 1706–1717. [CrossRef]

14.   Jiang, Q.; Qian, Y.; Ma, J.; Ma, X.; Cheng, Q.; Wei, F. User centric three-factor authentication protocol for cloud-assited wearable devices. *Int. J. Commun. Syst.* **2009**, *9*, e3900. [CrossRef]

15.   Madhusudhan, R.; Mittal, R.C. Dynamic id-based remote user password authentication schemes using smart cards: A review. *J. Netw. Comput. Appl.* **2012**, *35*, 1235–1248. [CrossRef]

16.   Wang, D.; He, D.; Wang, P.; Chu, C.-H. Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment. *IEEE Trans. Dependable Secur. Comput.* **2015**, *12*, 428–442. [CrossRef]

17.   Wang, D.; Wang, P. Two birds with one stone: Two-factor authentication with security beyond conventional bound. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 708–722. [CrossRef]

18.   Lamport, L. Password authentication with insecure communication. *Commun. ACM* **1981**, *24*, 770–772. [CrossRef]

19.   Hwang, T.; Chen, Y.; Laih, C.S. Non-interactive password authentication without password tables. In Proceedings of the 1990 IEEE Region 10 Conference on Computer and Communication Systems, Hong Kong, China, 24–27 September 1990.

20.   Lin, L.C.; Hwang, M.S.; Li, L.H. A new remote user authentication scheme for multi-server architecture. *Future Gener. Comput. Syst.* **2003**, *19*, 13–22. [CrossRef]

21.   Li, L.; Lin, L.; Hwang, M.S. A remote password authentication scheme for multi-server architecture using neural networks. *IEEE Trans. Neural Netw.* **2001**, *12*, 1498–1504. [PubMed]

22.   Liao, Y.P.; Wang, S.S. A secure dynamic ID based remote user authentication scheme for multi-server environment. *Comput. Stand. Interfaces* **2009**, *31*, 24–29. [CrossRef]

23.   Hsiang, C.; Shih, W.K. Improvement of the secure dynamic ID based remote user authentication scheme for multi-server environment. *Comput. Stand. Interfaces* **2009**, *31*, 1118–1123. [CrossRef]

24.   Martínez-Peláez, R.; Rico-Novella, F.; Satizábal, C.; Pomykala, J. Efficient and secure dynamic ID-based remote user authentication scheme with session key agreement for multi-server environment. *Int. J. Netw. Secur. Its Appl.* **2010**, *2*, 106–116. [CrossRef]

25.   Kim, M.; Park, N.; Won, D. Security Improvement on a Dynamic ID-Based Remote User Authentication Scheme with Session Key Agreement for Multi-server Environment. In *Computer Applications for Security, Control and System Engineering*; Springer: Berlin/Heidelberg, Germany, 2012.

26.   Li, X.; Xiong, Y.P.; Ma, J.; Wang, W.D. An efficient and security dynamic identity based authentication protocol for multi-server architecture using smartcards. *J. Netw. Comput. Appl.* **2012**, *35*, 763–769. [CrossRef]

27.   Sood, S.K.; Sarje, A.K.; Singh, K. A secure dynamic identity based authentication protocol for multiserver architecture. *J. Netw. Comput. Appl.* **2011**, *34*, 609–618. [CrossRef]

28.   Xue, K.; Hong, P.; Ma, C. A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture. *J. Comput. Syst. Sci.* **2014**, *80*, 195–206. [CrossRef]

29.   Challa, S.; Das, A.K.; Gope, P.; Kumar, N.; Wu, F.; Vasilakos, A.V. Design and analysis of authenticated key agreement scheme in cloud-assisted cyber-physical systems. *Future Gener. Comput. Syst.* **2018**. [CrossRef]

30.   Zhou, L.; Li, X.; Yeh, K.H.; Su, C.; Chiu, W. Lightweight IoT-based authentication scheme in cloud computing circumstance. *Future Gener. Comput. Syst.* **2019**, *91*, 244–251. [CrossRef]

31.   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]

32. Wu, F.; Xu, L.; Kumari, S.; Li, X.; Shen, J.; Raymond-Choo, K.K.; 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**, *89*, 72–85. [CrossRef]

33. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In *Advances in Cryptology*; Wienner, M., Ed.; Springer: Berlin, Germany, 1999; pp. 388–397.

34. Messerger, T.S.; Dabbish, E.A.; Sloan, R.H. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* **2002**, *51*, 541–552. [CrossRef]