


Article

Rotating behind Security: A Lightweight Authentication Protocol Based on IoT-Enabled Cloud Computing Environments

Tsu-Yang Wu ¹, Qian Meng ¹, Saru Kumari ² and Peng Zhang ^{1,*}

¹ College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; wutsuyang@gmail.com (T.-Y.W.); MQ15753683129@163.com (Q.M.)

² Department of Mathematics, Chaudhary Charan Singh University, Meerut 250004, India; saryusiirohi@gmail.com

* Correspondence: pengzhang_skd@sdust.edu.cn

Abstract: With the rapid development of technology based on the Internet of Things (IoT), numerous IoT devices are being used on a daily basis. The rise in cloud computing plays a crucial role in solving the resource constraints of IoT devices and in promoting resource sharing, whereby users can access IoT services provided in various environments. However, this complex and open wireless network environment poses security and privacy challenges. Therefore, designing a secure authentication protocol is crucial to protecting user privacy in IoT services. In this paper, a lightweight authentication protocol was designed for IoT-enabled cloud computing environments. A real or random model, and the automatic verification tool ProVerif were used to conduct a formal security analysis. Its security was further proved through an informal analysis. Finally, through security and performance comparisons, our protocol was confirmed to be relatively secure and to display a good performance.

Keywords: IoT; cloud computing; authentication protocol; formal security analysis



Citation: Wu, T.-Y.; Meng, Q.; Kumari, S.; Zhang, P. Rotating behind Security: A Lightweight Authentication Protocol Based on IoT-Enabled Cloud Computing Environments. *Sensors* **2022**, *22*, 3858. <https://doi.org/10.3390/s22103858>

Academic Editors: Muhammad Naveed Aman, Shehzad Ashraf Chaudhry and Chien-Ming Chen

Received: 8 April 2022
Accepted: 16 May 2022
Published: 19 May 2022

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



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Internet of things (IoT) [1–4] is the “Internet connected by all things”. It is the combination of networks and various sensing devices and compose a huge network that can interconnect users and everything whenever and wherever. The emergence of IoT has driven the development of many industries, such as transportation, agriculture, medical treatment, and artificial intelligence [5–7]. It has since made significant advancements and can connect various devices with limited resources, and massive amounts of data can be shared through the Internet.

Cloud computing [8,9] can connect a large number of resources, such as computation, software, and storage resources, to compose a large virtually shared resource pool [10]. Its core idea is to continuously lower the processing load of user terminals by increasing the processing capacity of the “cloud”, allowing users to exploit the “cloud’s” strong computing processing capacity on demand. With cloud computing, users can access applications on any device that can connect to the Internet [11]. The progress of cloud computing technology has penetrated all aspects of people’s lives and significantly increased the level of convenience during daily life.

In real life, the resource, computing, and communication capabilities of IoT devices are limited. To address these limitations, cloud computing, as a key technology, provides an efficient platform for effectively analyzing, managing, and storing the data generated by IoT devices. Mobile devices allow users to access the cloud server resources at any time from any location. Figure 1 shows the architecture of IoT-enabled cloud computing. This architecture has three entities: control server, user, and cloud server. The cloud server provides the services requested by users conveyed through user IoT devices. The control server is a trusted organization that authorizes users and the cloud server and creates system parameters during the registration phase. In addition, when users intend to obtain

the cloud server service, the control server monitors the authentication process, and with help from the control server, the three parties can consult a session key, which the user uses to obtain and enjoy the service of the cloud server.

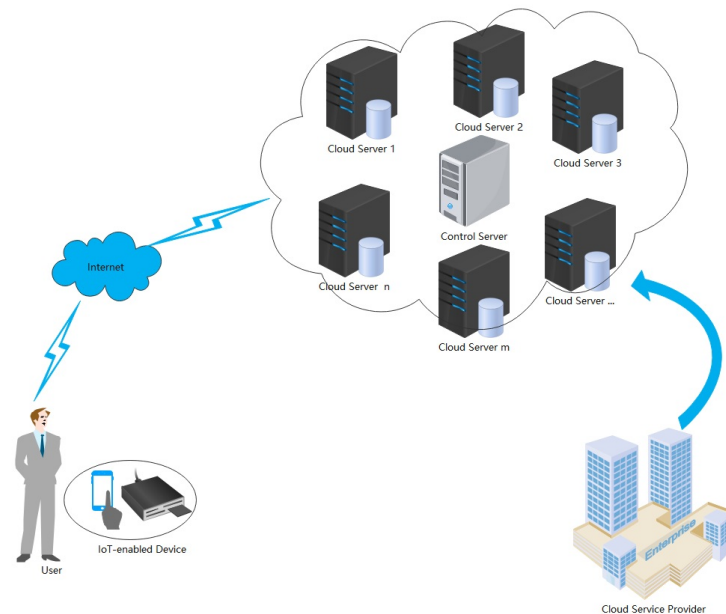


Figure 1. Architecture of IoT-enabled cloud computing.

Motivation

In IoT-enabled cloud computing environments [12–15], information is transmitted to the public channel, which is open and unprotected, and users are vulnerable to attackers when obtaining services, resulting in privacy data disclosure issues. Therefore, when users want to obtain cloud services, they must complete identity authentication and establish a key to protect the information from disclosure and tampering. At present, some scholars also use quick response (QR) codes [16] to solve these problems. Many scholars proposed authentication protocols [13,17–19] for this environment. However, these protocols typically have security problems, such as an inability to provide perfect forward security, suffering from man-in-the-middle (MITM) and temporary value disclosure attacks. In addition, the power of IoT devices is limited, and reducing the calculation of such devices is necessary.

In this paper, we designed a lightweight authentication protocol to solve the above problems. Both formal and informal security analyses were conducted to verify the security of our protocol. Through security and performance comparisons, our protocol demonstrated a good performance and satisfied the security requirements in IoT-enabled cloud computing environments.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 presents our protocol in detail. Section 4 introduces a safety analysis. Section 5 presents some security and performance comparisons, and Section 6 presents our conclusions.

2. Related Work

This section reviews authentication and key agreement (AKA) protocols [13,17–26] applied in IoT, cloud computing, and IoT-enabled cloud computing environments. A summary of existing protocols is shown in Table 1. Turkanovic et al. [22] designed an AKA scheme for IoT environments, which are dedicated to the identity authentication of users in wireless sensor networks. However, Wazid et al. [23] testified that Turkanovic et al.'s scheme [22] was unable to prevent insider and user impersonation attacks. Subsequently, Wu et al. [24] designed an AKA protocol and declared that it could prevent many common attacks. Unfortunately,

Sadri et al. [27] indicated that Wu et al.'s protocol was unable to resist sensor capture and denial of service attacks (DoS) and was, thus, unable to provide perfect forward security.

Table 1. A summary of authentication protocols.

Protocols	Advantages	Shortcomings
Turkanovic et al. [22]	(1) Provides user anonymity (2) Can resist offline password-guessing attacks	(1) Cannot resist insider attacks (2) Cannot resist user impersonation attacks
Wazid et al. [23]	(1) Can resist user impersonation attacks (2) Provides user anonymity (3) Provides perfect forward security	-
Wu et al. [24]	(1) Can resist temporary value disclosure attacks (2) Can resist offline password-guessing attacks	(1) Cannot resist sensor capture attacks (2) Cannot resist denial of service attacks (3) Cannot provide perfect forward security
Tsai and Lo [25]	(1) Can resist temporary value disclosure attacks (2) Provides perfect forward security	(1) Cannot resist server impersonation attacks
Irshad et al. [26]	(1) Can resist user impersonation attacks Provides perfect forward security	(1) Lacks user registration and revocation phases
Amin et al. [13]	(1) Can resist temporary value disclosure attacks (2) Can resist insider attacks	(1) Cannot prevent insider attacks (2) Cannot resist impersonation attacks
Martinez et al. [17]	(1) Can resist user impersonation attacks (2) Can resist offline password-guessing attacks (3) Provides user anonymity	(1) Cannot prevent impersonation attacks (2) Cannot resist session key exposure attacks (3) Cannot achieve mutual authentication
Zhou et al. [18]	(1) Provides user anonymity (2) Can achieve mutual authentication (3) Can resist insider attacks	(1) Cannot prevent replay attacks (2) Cannot prevent impersonation attacks (3) Cannot prevent temporary value disclosure attacks (4) cannot provide perfect forward security
Kang et al. [19]	(1) Can resist impersonation attacks (2) Can achieve mutual authentication	(1) Cannot resist offline password-guessing attacks

Tsai and Lo [25] designed an anonymous AKA scheme for cloud computing environments. They use bilinear pairing to design the scheme, and without the assistance of a control server, users can directly obtain the services of the distributed cloud server. However, He et al. [28] proved that their scheme cannot resist server impersonation attacks. Irshad et al. [26] designed an AKA scheme using a bilinear pairing method. Unfortunately, Xiong et al. [29] verified that Irshad et al.'s [26] lacked user registration and revocation phases. Xiong et al. [29] designed an enhanced scheme and claimed that it can prevent many common attacks.

Amin et al. [13] also designed a protocol applicable to distributed cloud computing environments. However, Challa et al. [30] testified that their protocol cannot prevent insider and impersonation attacks. Martinez et al. [17] designed a lightweight AKA scheme for cloud computing environments. Unfortunately, Yu et al. [31] determined that the scheme cannot prevent impersonation and session key exposure attacks or achieve mutual authentication. Zhou et al. [18] proposed a lightweight AKA scheme. However, Wang et al. [32] proved that their protocol cannot provide perfect forward security and cannot prevent replay, impersonation, and temporary value disclosure attacks. Kang et al. [19] designed a protocol suitable for IoT-enabled cloud computing environments, which supports the authentication of IoT devices. However, Huang et al. [33] verified that it was unable to resist offline password-guessing attacks.

3. The Proposed Protocol

This section introduces our protocol. It includes three phases: (1) user registration, (2) cloud server registration, and (3) login and authentication. The following subsections describe each in detail. Table 2 lists the symbols used in the protocol.

Table 2. Notations.

Notations	Meanings
S_j	The j th cloud server
SID_j	The S_j 's identity
U_i	The i th user
ID_i	U_i 's identity
PW_i	U_i 's password
B_i	U_i 's biological information
HPW_i	U_i 's pseudo password
SC	Smart card
CS	Control server
ID_{CS}	CS 's identity
x	The secret key of CS
TID_i	U_i 's pseudo identity
QID_j	S_j 's pseudo identity
$h(\cdot)$	Hash function
$Gen(\cdot), Rep(\cdot)$	Fuzzy extraction function
τ_i, σ_i	Two parameters generated by the fuzzy extractor [34], where τ_i is public and σ_i is private.
TS_1, TS_2, TS_3, TS_4	Timestamps

3.1. System Model

Our IoT-enabled cloud computing model includes three entities, namely user, cloud server, and control server. The information exchange between each entity is shown in Figure 2.

- (1) User: The user can use IoT devices to obtain cloud server services. We allow the user to be an untrusted entity, which means that they may be a legitimate user but may obtain services or launch attacks maliciously.
- (2) Cloud server: The cloud server provides the services requested by users conveyed through user IoT devices. It is a semi-trusted entity, in the sense that it may misbehave on its own but does not conspire with either of the participants.
- (3) Control server: The control server is responsible for registering users and cloud server, assisting users and cloud server in completing authentication and in establishing a session key in the login and authentication phase. It is a semi-trusted entity, in the sense that it may misbehave on its own but does not conspire with either of the participants.

The purpose of our protocol is to realize mutual authentication and to establish a session key between the user and cloud server with the help of the control server. Figure 2

shows the exchange of information. The specific process is referred to in Section 3.4 (Login and Authentication Phase).

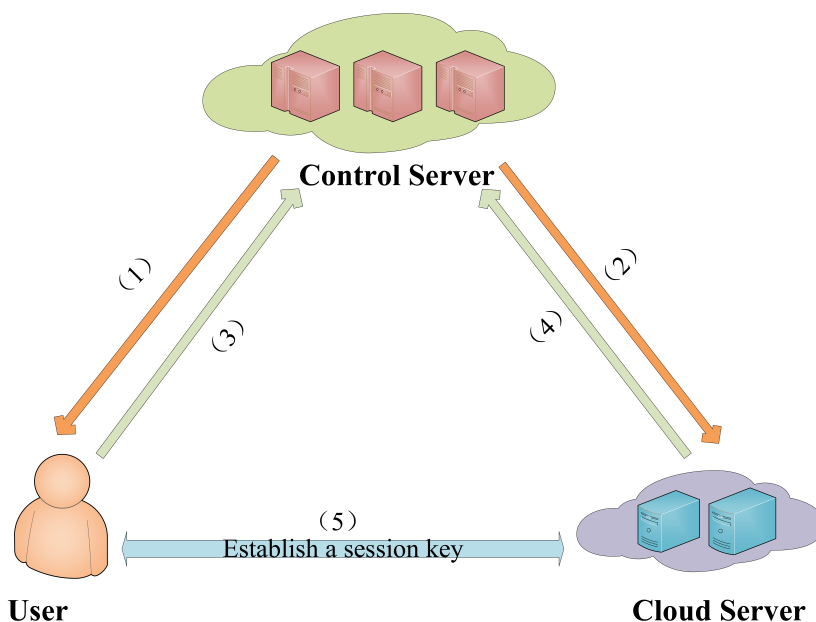


Figure 2. Information exchange process.

3.2. User Registration Phase

At this phase, U_i registers with CS as a legal user. The user transmits the parameters calculated by themselves to CS via a secure channel and finally obtains the smart card issued by CS. Figure 3 detail the process. The specific process is as follows:

- (1) U_i chooses ID_i , PW_i , and B_i ; calculates $Gen(B_i) = (\sigma_i, \tau_i)$ and $HPW_i = h(PW_i \parallel \sigma_i)$; and then sends $\{ID_i, HPW_i\}$ to control server CS through a secure channel.
- (2) CS checks U_i 's identity. If the identity is new, CS selects a random value n_i and computes $TID_i = h(ID_i)$, $A_1 = h(ID_{CS} \parallel HPW_i) \oplus (n_i \oplus K_j)$, stores $\{TID_i, HPW_i\}$ in the database, stores $\{A_1, ID_{CS}\}$ in smart card SC, and then sends SC to U_i through a secure channel.
- (3) After receiving message $\{A_1, ID_{CS}\}$ sent by CS, U_i calculates $A_2 = h(ID_i \parallel HPW_i)$ and then stores $\{A_2, Gen(\cdot), Rep(\cdot), \tau_i\}$ in SC.

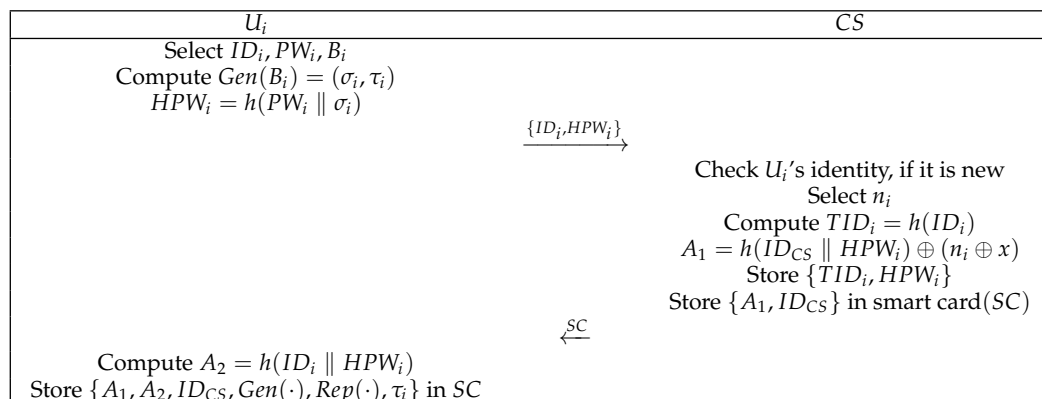


Figure 3. User registration phase.

3.3. Cloud Server Registration Phase

At this phase, cloud server S_j needs to register with CS as a legal entity. It sends its own parameters to CS via a secure channel, obtains the parameters calculated by the

CS, and stores them in its own memory. Figure 4 shows specific the process. The specific process is as follows:

- (1) S_j selects its identity SID_j and random number n_j and then sends $\{SID_j, n_j\}$ to CS through a secure channel.
- (2) CS checks the identity of S_j . If S_j is unregistered, then CS selects a pseudo identity QID_j for S_j , calculates $A_3 = h(SID_j \parallel K_j \oplus n_j)$, and stores $\{QID_j, n_j\}$ in its memory. Then, CS sends $\{QID_j, A_3\}$ to S_j through a secure channel.
- (3) S_j calculates $A_3^* = A_3 \oplus SID_j$ and stores $\{A_3^*, QID_j\}$ in its memory.

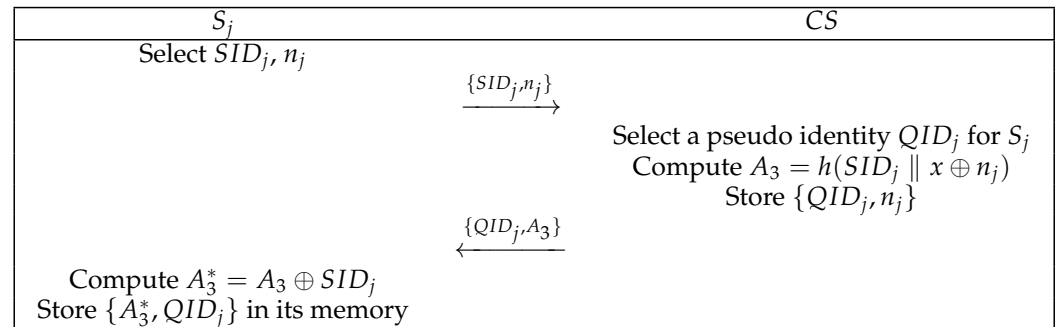


Figure 4. Cloud server registration phase.

3.4. Login and Authentication Phase

At this phase, the control server CS verifies the identity of the user U_i and cloud server S_j . After verification, the three establish a common session key for future communication. The specific process is shown in the Figure 5. The specific process is as follows:

- (1) U_i inputs ID_i and PW_i ; imprints B_i ; computes $Rep(B_i, \tau_i) = \sigma_i$, $HPW_i = h(PW_i \parallel \tau_i)$, $A_2' = h(ID_i \parallel HPW_i)$; and checks the legitimacy of U_i 's identity by verifying $A_2' \stackrel{?}{=} A_2$. If this is valid, U_i then chooses a random value r_i and timestamp TS_1 and computes $(n_i \oplus x) = A_1 \oplus h(ID_{CS} \parallel HPW_i)$, $B_1 = r_i \oplus h(ID_{CS} \parallel HPW_i \oplus SID_j)$, $B_2 = SID_j \oplus h(ID_{CS} \parallel HPW_i)$, and $B_3 = h(TID_i \parallel ID_{CS} \parallel n_i \oplus x) \oplus HPW_i$. Subsequently, $M_1 = \{TID_i, A_1, B_1, B_2, B_3, TS_1\}$ is sent to S_j through an open channel.
- (2) After receiving U_i 's message, S_j checks timestamp $|TS_1 - TS_c| \leq \Delta T$. If the timestamp is valid, S_j then selects a random number r_j and timestamp TS_2 . S_j calculates $A_3 = SID_j \oplus A_3^*$, $B_4 = r_j \oplus h(A_3 \parallel SID_j)$, and $B_5 = h(r_j \parallel A_3 \parallel SID_j)$ and then sends message $M_2 = \{M_1, QID_j, B_4, B_5, TS_2\}$ to CS through an open channel.
- (3) After receiving M_2 , CS checks timestamp $|TS_2 - TS_c| \leq \Delta T$. If the verification passes, CS finds HPW_i according to TID_i ; computes $SID_j = B_2 \oplus h(ID_{CS} \parallel HPW_i)$, $r_i = B_1 \oplus h(ID_{CS} \parallel HPW_i \oplus SID_j)$, and $B_3' = h(TID_i \parallel ID_{CS} \parallel n_i \oplus x) \oplus HPW_i$; and verifies U_i 's identity by checking $B_3' \stackrel{?}{=} B_3$. If valid, CS then indexes n_j according to the value of QID_j ; computes $A_3 = h(SID_j \parallel x \oplus n_j)$, $r_j = B_4 \oplus h(A_3 \parallel SID_j)$, and $B_5' = h(r_j \parallel A_3 \parallel SID_j)$; and checks $B_5' \stackrel{?}{=} B_5$. If valid, CS then selects r_k, TS_3 computes $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$, $B_6 = (r_i \oplus HPW_i) \oplus A_3$, $B_7 = h(A_3 \parallel r_j \parallel SID_j) \oplus r_k$, $B_8 = h(r_j \parallel r_k \parallel SK \parallel TS_3)$, $(n_i \oplus x) = A_1 \oplus h(ID_{CS} \parallel HPW_i)$, $B_9 = h(n_i \oplus x \parallel SID_j) \oplus r_j$, and $B_{10} = h(HPW_i \parallel r_i) \oplus r_k$, $B_{11} = h(SK \parallel n_i \oplus x \parallel r_k \parallel r_j)$ and sends message $M_3 = \{B_6, B_7, B_8, B_9, B_{10}, B_{11}, TS_3\}$ to S_j through an open channel.
- (4) After receiving M_3 , the cloud server checks the timestamp $|TS_3 - TS_c| \leq \Delta T$. If the timestamp is valid, S_j then computes $(r_i \oplus HPW_i) = B_6 \oplus A_3$, $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$, and $B_8' = h(r_j \parallel r_k \parallel SK \parallel TS_3)$, and checks $B_8' \stackrel{?}{=} B_8$. If true, S_j sends message $M_4 = \{B_9, B_{10}, TS_4\}$ to U_i through an open channel.
- (5) U_i checks timestamp $|TS_4 - TS_c| \leq \Delta T$. If the verification passes, U_i then computes $r_j = h(n_i \oplus x \parallel SID_j) \oplus B_9$, $r_k = h(HPW_i \parallel r_i) \oplus B_{10}$, $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel$

- SID_j), and $B'_{11} = h(SK \parallel n_i \oplus x \parallel r_k \parallel r_j)$ and checks $B'_{11} \stackrel{?}{=} B_{11}$. If the verification passes, U_i then computes $B_{12} = h(SK \parallel r_j)$ and sends $M_5 = \{B_{12}\}$ to S_j .
- (6) S_j computes $B'_{12} = h(SK \parallel r_j)$ and checks $B'_{12} \stackrel{?}{=} B_{12}$. If the verification passes, then S_j stores SK for future communication.

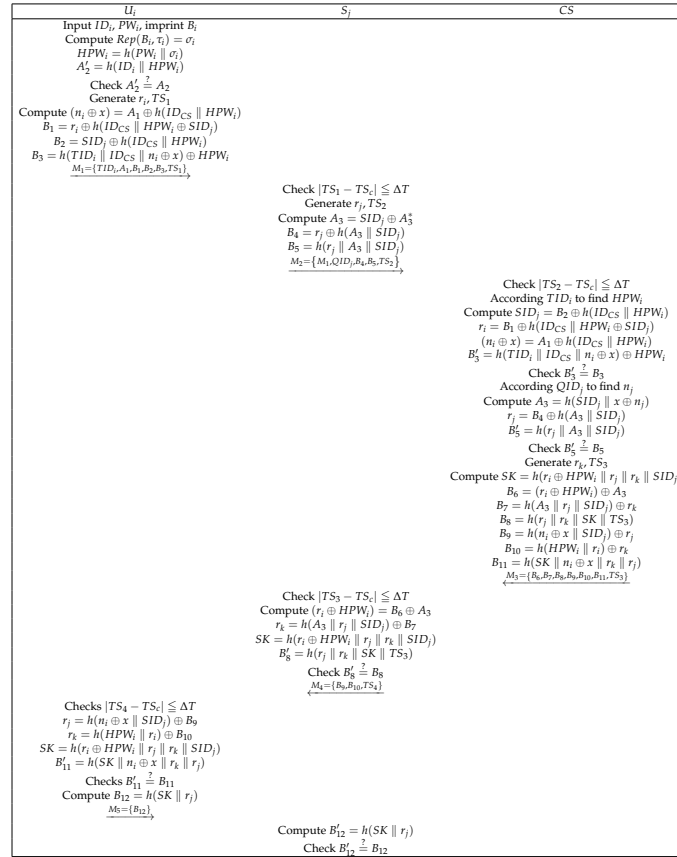


Figure 5. Login and authentication phase.

4. Security Analysis

This section presents an informal security analysis and describes a formal analysis using ProVerif and the real or random (ROR) model. The subsections introduce these topics.

4.1. Attacker Model

We define the attacker's ability based on the C-K model [35], which is an extension of the D-Y model [36]. The following features of an attacker \mathcal{A} are defined:

- (1) \mathcal{A} is assumed to be capable of blocking, modifying, and eavesdropping on messages transmitted on the open channel. It has complete control over communications between the various participants.
- (2) \mathcal{A} can be a malicious insider on the control server and can obtain the content stored in the control server by the user or cloud server.
- (3) \mathcal{A} can disclose the established session key, long-term key, and session state.
- (4) \mathcal{A} can guess the user's password or identity, but \mathcal{A} is unable to guess the user's identity or password simultaneously in polynomial time.
- (5) \mathcal{A} may extract the information of a user's SC using power analysis.

4.2. Formal Security Analysis

We use the ROR model and the automated verification tool ProVerif to conduct a formal security analysis to testify that the protocol is secure and correct.

4.2.1. ROR Model

The protocol security is demonstrated using the ROR model [4,37]. The security is verified by calculating the probability of session key SK .

The protocol comprises three parties: user, cloud server, and control server. In this model, $\Pi_{U_i}^x$, $\Pi_{S_j}^y$, and Π_{CS}^z are the x th user, y th cloud server, and z th control server, respectively. Suppose attacker \mathcal{A} 's query capabilities include the following: $Z = \Pi_{U_i}^x, \Pi_{S_j}^y$, and Π_{CS}^z .

Execute(Z): Assuming an attacker \mathcal{A} executes the query, they can capture messages on the open channel.

Send(Z, M): Assuming an attacker \mathcal{A} executes the query, they transfer M to Z and receive an answer from Z .

Hash(*string*): Suppose an attacker \mathcal{A} executes the query; they enter a string and obtain a hash value.

Corrupt(Z): Assuming an attacker \mathcal{A} executes the query, they obtain the private value of an entity, for example, a long-term key and temporary information of the user's SC.

Test(Z): Assume that an attacker \mathcal{A} executes the query and tosses a coin C into the air. If C equals 1, \mathcal{A} obtains SK . Otherwise, \mathcal{A} obtains a string.

Theorem 1. *If \mathcal{A} executes queries *Execute*(Z), *Send*(Z, M), *Hash*(*string*), *Corrupt*(Z), and *Test*(Z), the probability P of \mathcal{A} cracking the protocol is $Adv_{\mathcal{A}}^P(\xi) \leq q_{send}/2^{l-2} + 3q_{hash}^2/2^{l-1} + 2\max\{c' \cdot q_{send}^{s'}, q_{send}/2^l\}$. Here, q_{send} refers to the numbers of times the queries executed, q_{hash} is the execution time of the hash function, l is the bit length of biological information [38], and c' and s' are two constants.*

Proof. The ROR model played $GM_0, GM_1, GM_2, GM_3, GM_4$. $Succ_{\mathcal{A}}^{GM_i}(\xi)$ is the probability that \mathcal{A} can win GM_0 – GM_4 . The following are the specific query steps in the game: GM_0 : GM_0 represents the first round of the game, which starts by flipping C . GM_0 cannot execute any queries; hence, the probability that \mathcal{A} can break P is as follows:

$$Adv_{\mathcal{A}}^P(\xi) = |2Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)] - 1|. \quad (1)$$

GM_1 : GM_1 is for the GM_0 -added *Execute*(Z) operation, and \mathcal{A} can be used only when GM_1 intercepts the messages M_1 – M_5 transmitted over the open channel. Then, because the values of HPW_i, r_i, r_j, r_k and SID_j cannot be obtained, \mathcal{A} cannot obtain the session key through the *Test*(Z) query. Thus, GM_1 's probability is the same as GM_0 .

$$Pr[Succ_{\mathcal{A}}^{GM_1}(\xi)] = Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)]. \quad (2)$$

GM_2 : GM_2 extends GM_1 by adding the *Send*(Z, M) query. The probability of GM_2 is calculated using Zipf's law [39].

$$|Pr[Succ_{\mathcal{A}}^{GM_2}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_1}(\xi)]| \leq q_{send}/2^l. \quad (3)$$

GM_3 : GM_3 is for the GM_2 -added *Hash*(*string*) operation and deleted *Send*(Z, M) operation. GM_3 's probability can be obtained using the birthday paradox.

$$|Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_2}(\xi)]| \leq q_{hash}^2/2^{l+1}. \quad (4)$$

GM_4 : In GM_4 , a security analysis on two events is conducted to testify the security of the session key. (1) \mathcal{A} obtains CS 's long-term key x ; (2) \mathcal{A} obtains the temporary information. This demonstrates that our protocol can guarantee perfect forward security and prevent temporary information disclosure attacks.

(1) Perfect forward security: \mathcal{A} with Π_{CS}^z to obtain x of CS or use $\Pi_{U_i}^x, \Pi_{S_j}^y$ to obtain private values.

- (2) Temporary information disclosure attack: \mathcal{A} utilizes $\Pi_{U_i}^x$, $\Pi_{S_j}^y$ or Π_{CS}^Z to obtain the random number of three entities.

For the first case, even if \mathcal{A} obtains x or some private values, they cannot calculate HPW_i, r_i, r_j, r_k , or SID_j . Therefore, \mathcal{A} cannot calculate SK , where $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$. For the second case, even if \mathcal{A} obtains r_i but HPW_i, r_j, r_k and SID_j are private, SK is incalculable. Similarly, even if \mathcal{A} can obtain r_j or r_k , SK is also incalculable. Thus, the probability of GM_4 is obtained:

$$|Pr[Succ_{\mathcal{A}}^{GM_4}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)]| \leq q_{send}/2^l + q_{hash}^2/2^{l+1}. \quad (5)$$

GM_5 : In GM_5 , \mathcal{A} queries the parameters $\{A_1, A_2, ID_{CS}, Gen(\cdot), Rep(\cdot), \tau_i\}$ in the smart card by executing $Corrupt(Z)$. This proves that our protocol can protect against offline password-guessing attacks. \mathcal{A} attempts to guess $A_2 = h(ID_i \parallel HPW_i)$, where $HPW_i = h(PW_i \parallel \tau_i)$. However, ID_i and HPW_i are private. The probability that \mathcal{A} can guess l bit of biological information is $1/2^l$. From Zipf's law [39], when $q_{send} \leq 106$, the probability that \mathcal{A} can guess the password is more than $1/2$. Thus, the probability of GM_5 can be obtained:

$$|Pr[Succ_{\mathcal{A}}^{GM_5}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_4}(\xi)]| \leq \max\{c' \cdot q_{send}^{s'}, q_{send}/2^l\} \quad (6)$$

GM_6 : GM_6 confirms that the protocol can prevent impersonation attacks. \mathcal{A} queries $h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$, and the game ends. Therefore, the probability of GM_6 can be obtained:

$$|Pr[Succ_{\mathcal{A}}^{GM_6}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_5}(\xi)]| \leq q_{hash}^2/2^{l+1}. \quad (7)$$

Because \mathcal{A} 's probability of success is the same as that of failure (i.e., (1)–(2)), \mathcal{A} 's probability of obtaining the session key is

$$Pr[Succ_{\mathcal{A}}^{GM_6}(\xi)] = 1/2. \quad (8)$$

From all these formulas,

$$\begin{aligned} 1/2 Adv_{\mathcal{A}}^{\mathcal{P}}(\xi) &= |Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)] - 1/2| \\ &= |Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_6}(\xi)]| \\ &= |Pr[Succ_{\mathcal{A}}^{GM_1}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_6}(\xi)]| \\ &\leq \sum_{i=0}^5 |Pr[Succ_{\mathcal{A}}^{GM_{i+1}}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_i}(\xi)]| \\ &= q_{send}/2^{l-1} + 3q_{hash}^2/2^l + \max\{c' \cdot q_{send}^{s'}, q_{send}/2^l\} \end{aligned} \quad (9)$$

Consequently, we obtain

$$Adv_{\mathcal{A}}^{\mathcal{P}}(\xi) \leq q_{send}/2^{l-2} + 3q_{hash}^2/2^{l-1} + 2\max\{c' \cdot q_{send}^{s'}, q_{send}/2^l\}. \quad (10)$$

□

4.2.2. ProVerif

ProVerif [40,41] is a powerful and appropriate tool for analyzing and verifying protocol security. We use it to verify our protocol's security.

- (1) Some functions and queries are also defined, as shown in Figure 6a,b.
- (2) Figure 6c shows the defined events and queries. Among them, we define eight queries. The first three queries prove the session key's security, while the other five queries prove the protocol's correctness. In addition, we also defined eight events. Event UserStarted() indicates that U_i begins authentication, event UserAuthenticated() indicates that U_i

- successfully authenticated, event ControlServerAcUser() represents CS authenticating U_i successfully, event ControlServerAcCloudServer() represents CS authenticating S_j successfully, event CloudServerAcControlServer() indicates that S_j successfully authenticates CS, event UserAcControlServer() represents U_i authenticating CS successfully, event UserAcCloudServer() represents U_i authenticating S_j successfully, and event CloudServerAcUser() represents CS authenticating U_i successfully.
- (3) Figure 7a–c shows U_i 's, S_j 's, and CS's processes, respectively. Finally, Figure 8 presents the results. The first three results demonstrate that attackers cannot obtain SK, and the last five outcomes demonstrate that the protocol is correct and reasonable. Therefore, our protocol can successfully pass the verification of ProVerif and prevent common attacks.

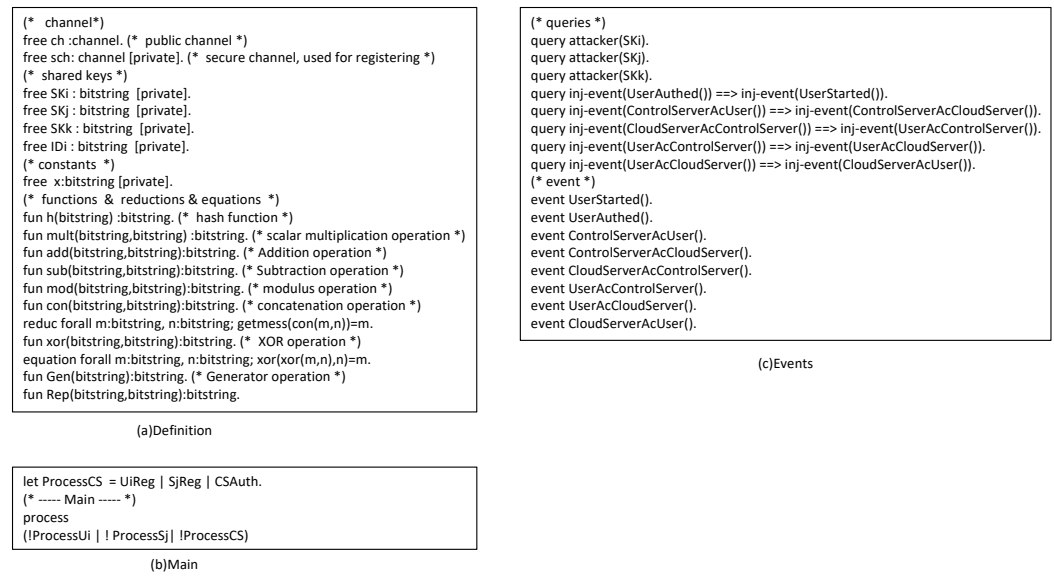


Figure 6. Definitions.

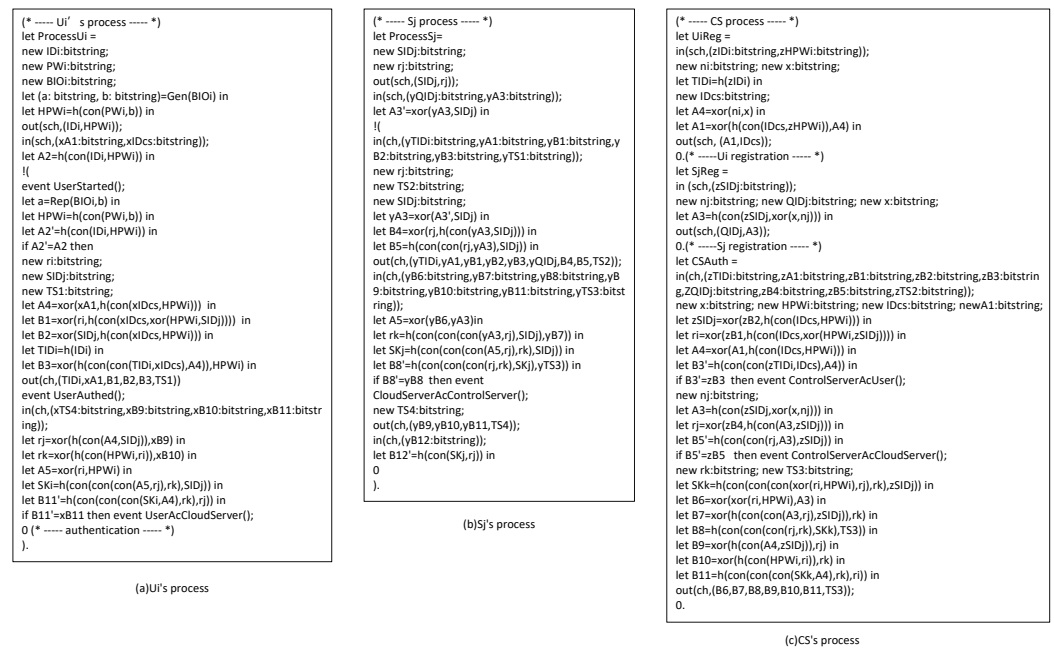


Figure 7. Process.

```

Verification summary:
Query not attacker(SKi[]) is true.
Query not attacker(SKj[]) is true.
Query not attacker(SKk[]) is true.
Query inj-event(UserAuthenticated) ==> inj-event(UserStarted) is true.
Query inj-event(ControlServerAcUser) ==> inj-event(ControlServerAcCloudServer) is true.
Query inj-event(CloudServerAcControlServer) ==> inj-event(UserAcControlServer) is true.
Query inj-event(UserAcControlServer) ==> inj-event(UserAcCloudServer) is true.
Query inj-event(UserAcCloudServer) ==> inj-event(CloudServerAcUser) is true.

```

Results

Figure 8. Results.

4.3. Informal Security Analysis

In this subsection, an informal analysis is adopted to demonstrate the common security requirements of the proposed protocol.

4.3.1. Man-in-the-Middle Attacks

\mathcal{A} computes SK by intercepting messages on the open channel. Let us suppose that message M_1 is intercepted and \mathcal{A} attempts to calculate $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$ but they cannot obtain the values of ID_{CS}, HPW_i, SID_j . Therefore, \mathcal{A} cannot use the message $\{B_1, B_2, B_4, B_7\}$ on the open channel to calculate r_i, r_j, r_k, B_3, B_5 ; change any values; or successfully pass the authentication of CS ; thus, they cannot successfully calculate SK . Consequently, the proposed protocol can guard against MITM attacks.

4.3.2. Insider Attacks

Case one: Assume that a malicious attacker \mathcal{A} obtains $\{QID_j, n_j, TID_i, HPW_i\}$ stored in the CS database. They use the message on the open channel to compute $r_i = B_1 \oplus h(ID_{CS} \parallel HPW_i \oplus SID_j)$. However, \mathcal{A} cannot obtain the values of ID_{CS}, SID_j , and thus, r_i cannot be calculated. Similarly, because \mathcal{A} cannot obtain the values of A_3, SID_j , \mathcal{A} cannot calculate $r_j = B_4 \oplus h(A_3 \parallel SID_j)$ and $r_k = h(HPW_i \parallel r_i) \oplus B_{10}$. Therefore, the session key cannot be computed using \mathcal{A} . Therefore, our protocol can prevent insider attacks.

Case two: Assume that the attacker \mathcal{A} is an insider of the cloud server and obtains the information A_3^*, QID_j stored in it. They then try to intercept the information on the open channel and to calculate the session key $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$. They intercepted B_4 and tried to calculate $r_j = B_4 \oplus h(A_3 \parallel SID_j)$ but cannot calculate the value of A_3 and thus cannot obtain the value of r_j . Similarly, \mathcal{A} attempts to intercept B_6 and B_7 to calculate $(r_i \oplus HPW_i) = B_6 \oplus A_3$, and $r_k = h(A_3 \parallel r_j \parallel SID_j) \oplus B_7$. However, they cannot obtain the value of A_3 and thus cannot calculate $(r_i \oplus HPW_i)$ and r_k , so they cannot successfully calculate SK .

By analyzing these two situations, we can prove that our protocol can resist insider attacks.

4.3.3. DDoS Attacks

During the login and authentication phase, U_i sends service request message $M_1 = \{TID_i, A_1, B_1, B_2, B_3, TS_1\}$ to S_j . After S_j receives M_1 , whether the timestamp is valid is checked first. If the timestamp is valid, S_j performs the following calculation. Therefore, if attacker \mathcal{A} wants to launch DDoS attacks, it must be within a valid time, and it is not possible in this protocol to deny a service only by sending a huge service request. Therefore, the protocol is immune to this attack.

4.3.4. Masquerading Attacks

Case one: Attacker \mathcal{A} attempts to impersonate any legitimate user, cloud server, or control server. Suppose that \mathcal{A} obtains the information $\{QID_j, n_j, TID_i, HPW_i\}$ stored in CS and intercepts the messages $\{M_1, M_2, M_3, M_4\}$ on the public channel. \mathcal{A} wants to impersonate a legitimate U_i by calculating $B_3 = h(TID_i \parallel ID_{CS} \parallel n_i \oplus x) \oplus HPW_i$, but \mathcal{A} cannot obtain values of ID_{CS} and $(n_i \oplus x)$. Therefore, they cannot successfully calculate the

value of B_3 and cannot impersonate a legitimate user by changing B_3 to pass the verification of CS and thus cannot pretend to be a legitimate user.

Case two: Similarly, \mathcal{A} wants to impersonate a S_j through $B_5 = h(r_j \parallel A_3 \parallel SID_j)$ but cannot obtain values of r_j , A_3 and SID_j , so they cannot pass the verification of CS . Therefore, \mathcal{A} cannot successfully impersonate a legal S_j . It can be concluded that the proposed protocol can resist impersonation attacks.

To sum up, our protocol can resist masquerading attacks.

4.3.5. Identity Theft Attacks

Suppose that an attacker \mathcal{A} obtains the user's smart card and tries to impersonate a legitimate user to establish a session with the cloud server and the control server. They obtain $\{A_1, A_2, ID_{CS}, Gen(\cdot), Rep(\cdot), \tau_i\}$ and try to calculate the authentication value $B_3 = h(TID_i \parallel ID_{CS} \parallel n_i \oplus x) \oplus HPW_i$ by intercepting the information on the open channel. Because they cannot obtain PW_i, τ_i , they cannot calculate $HPW_i = h(PW_i \parallel \tau_i)$ and thus cannot successfully calculate B_3 and pass the authentication of CS . Therefore, our protocol can resist identity theft attacks.

4.3.6. Replay Attacks

According to our defined attacker model, an attacker \mathcal{A} can forward the intercepted message to the receiver on the open channel and prove that they are a legitimate entity if the receiver authenticates the message. However, each transmitted message has a timestamp. If \mathcal{A} transmits a previously intercepted message, the recipient rejects the request because of the invalid timestamp. Thus, the protocol is resistant to replay attacks.

4.3.7. Perfect Forward Secrecy

In our protocol, $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$. Case one: Suppose that an attacker \mathcal{A} can obtain x but SID_j and HPW_i cannot be computed and \mathcal{A} cannot obtain random numbers r_i, r_j , and r_k . Therefore, there is no way to calculate the current SK or the previous SK , so the proposed protocol can provide perfect forward security.

Case two: Assume that an attacker \mathcal{A} obtains the user's password PW_i to attack. Because the user's biological information B_i cannot be obtained, \mathcal{A} cannot compute HPW_i , where $HPW_i = h(PW_i \parallel \tau_i)$. Additionally, $\{r_i, r_j, r_k, SID_j\}$ is unknown. \mathcal{A} cannot successfully compute SK .

Case three: Assume that an attacker \mathcal{A} can obtain the private value A_3^* of a cloud server for an attack. Because the identity SID_j of the S_j cannot be obtained, \mathcal{A} cannot calculate A_3 , where $A_3 = h(SID_j \parallel x \oplus n_j)$. Furthermore, \mathcal{A} cannot calculate r_j and r_k ; here, $r_i = B_1 \oplus h(ID_{CS} \parallel HPW_i \oplus SID_j)$ and $r_k = h(A_3 \parallel r_j \parallel SID_j) \oplus B_7$. Additionally, HPW_i is unknown, and \mathcal{A} cannot successfully calculate SK .

Therefore, the proposed protocol can provide perfect forward security.

4.3.8. Session Key Disclosure Attacks

It is assumed that the attacker \mathcal{A} attempts to intercept the transmission of information on the open channel. Even if the attacker intercepts the messages $M_1 - M_5$, they cannot compute $r_j = B_4 \oplus h(A_3 \parallel SID_j)$, $r_k = h(A_3 \parallel r_j \parallel SID_j) \oplus B_7$, and $(r_i \oplus HPW_i) = B_6 \oplus A_3$ because they cannot obtain the values of HPW_i, SID_j, A_3 . Obviously, they cannot compute the session key $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$ by intercepting the information on the public channel. Therefore, our proposed protocol can resist session key disclosure attacks.

4.3.9. Mutual Authentication

In the login and authentication phase, CS verifies the user and cloud server through B_3 and B_5 , respectively, and B_8 and B_{11} are the values of U_i and S_j used to verify mutual identity, respectively. Although B_3 and B_5 are transmitted over the open channel, the values of $(n_i \oplus x)$, HPW_i, r_j , and A_3 cannot be obtained by an attacker \mathcal{A} . Similarly, B_8 and B_{11} are also transmitted over the open channel, but \mathcal{A} cannot obtain the values of r_k and SK , and

thus, the protocol cannot break by changing the authentication value. Hence, our protocol can provide mutual authentication.

4.3.10. Privacy and Anonymity

An attacker \mathcal{A} attempts to identify a user by intercepting messages on the open channel. However, in our proposed method, \mathcal{A} can only obtain U_i 's pseudo identity TID_i . Thus, \mathcal{A} cannot compute the user's real ID_i . Similarly, \mathcal{A} can only obtain the S_j 's pseudo identity QID_j . \mathcal{A} cannot determine the true identity of U_i and S_j based on the pseudo identity, which protects the privacy of U_i and S_j . Therefore, our protocol can provide privacy and anonymity.

4.3.11. Traceability and Non-Repudiation

When cloud server finds that U_i has bad behavior, it will report to CS, and CS will find the value of the user's HPW_i according to TID_i , which can be used to identify U_i . Therefore, once a user exhibits malicious behavior, CS can track the user, which ensures traceability. Since the transmitted message $M_1 = \{TID_i, A_1, B_1, B_2, B_3, TS_1\}$ contains the value of authenticating the user's identity B_3 , once a legitimate user exhibits bad behavior, CS will verify the user's identity according to $B_3 = h(TID_i \parallel ID_{CS} \parallel n_i \oplus x) \oplus HPW_i$. If the verification is passed, this indicates that the bad behavior is indeed sent by the user, and the user cannot deny it. Therefore, non-repudiation is guaranteed.

4.3.12. Integrity

Integrity is the guarantee that an attacker \mathcal{A} cannot change the transmitted information. Even if \mathcal{A} is able to successfully tamper with the information, the system will detect and discover that the information has been modified.

It is assumed that an attacker \mathcal{A} can intercept and tamper with the messages $\{M_1, M_2, M_3, M_4\}$ transmitted on the open channel. For example, \mathcal{A} intercepts and tampers with message M_1 , where $M_1 = \{TID_i, A_1, B_1, B_2, B_3, TS_1\}$. If \mathcal{A} tampers with TID_i , CS cannot retrieve HPW_i and the authentication is suspended. If \mathcal{A} tampers with A_1, B_1, B_2, B_3 , then CS calculates that B'_3 is not equal to the received $B_3 = h(TID_i \parallel ID_{CS} \parallel n_i \oplus x) \oplus HPW_i$, which indicates that the user is not legal or M_1 is tampered with, and the authentication is suspended. Similarly, if an attacker intercepts and tampers with $\{M_2, M_3, M_4\}$, all three entities will be checked accordingly. Therefore, the proposed protocol can ensure the integrity of information.

4.3.13. Confidentiality

From the Section 4.3.2 (Insider Attacks) and Section 4.3.4 (Masquerading Attacks), it can be seen that the attacker cannot obtain $SK = h(r_i \oplus HPW_i \parallel r_j \parallel r_k \parallel SID_j)$. Therefore, it can be seen that our protocol ensures confidentiality.

5. Security and Performance Comparison

We compared the protocols of Amin et al. [13], Martinez et al. [17], Zhou et al. [18], and Kang et al. [19] in terms of performance and security. The specific comparison results are described in the following subsection.

5.1. Security Comparison

This subsection compares the five protocols in terms of security. Specifically, \checkmark indicates that the security characteristics are met, and \times indicates that they are not met. In addition, S1–S8 are defined as follows: S1: Perfect forward secrecy; S2: Man-in-the-middle attack; S3: Mutual authentication; S4: Impersonation attack; S5: Replay attack; S6: Temporary value disclosure attack; S7: Offline password-guessing attack; and S8: Insider attack.

Table 3 lists the security results. From Table 3, Zhou et al.'s [18] scheme was unable to provide perfect forward security and cannot prevent replay, user and server impersonation,

and temporary value disclosure attacks. The protocol of Kang et al. [19] cannot resist offline password-guessing attacks; Amin et al.'s [13] protocol cannot prevent insider or impersonation attacks; and the protocol of Martinez et al. [17] was unable to prevent impersonation or replay attacks, or even enable mutual authentication. The proposed protocol can evidently prevent many common attacks.

Table 3. Comparisons of security.

Security Properties	[13]	[17]	[18]	[19]	Ours
S1	✓	✓	×	✓	✓
S2	×	✓	✓	✓	✓
S3	✓	×	✓	✓	✓
S4	×	×	×	✓	✓
S5	✓	×	×	✓	✓
S6	✓	✓	×	✓	✓
S7	✓	✓	✓	×	✓
S8	×	✓	✓	✓	✓

5.2. Performance Comparison

We calculated the time required by the user and server. To estimate the user's computing cost, we developed an app that uses the Java pairing library, signature library, and symmetric encryption/decryption function to calculate the running time of various operations. We used smart phones produced by different manufacturers to imitate the user. We ran various operations on the following mobile phones ten times and used the average value as the reference time. Table 4 lists the results of various operations on different mobile phones. D1 is a Huawei Mate 30 mobile phone with a harmony operating system, Huawei Kirin 990 processor, and 8G running memory; D2 is a Redmi Note 9 Pro mobile phone with an Android operating system, Qualcomm Snapdragon 750 g CPU, and 8G of RAM; and D3 represents a OnePlus phone with an Android operating system, Snapdragon 865 processor, and 8G running memory. Table 5 and Figure 9 present the comparative results of the client calculation costs of the five protocols. Table 5 shows that the protocol of Amin et al. [13] requires $9T_h$, the protocol of Martinez et al. [17] requires $11T_h + T_{de}$, Zhou et al.'s [18] protocol requires $10T_h$, Kang et al.'s protocol [19] requires $8T_h$, and our proposed protocol requires $T_m + 10T_h$. Although we are not as good as Amin et al. [13], Zhou et al. [18], and Kang et al. [19] in terms of performance, we are better than them in terms of security.

Table 4. The computational costs of complex operations.

Operations	Symbolic	D1 (ms)	D2 (ms)	D3 (ms)	Server (Cloud, Contorl)
Symmetric Decryption	T_{de}	0.04125	0.2	0.2	0.1347
Symmetric Encryption	T_{en}	0.2	0.0392	0.0591	4.7
Hash function	T_h	0.00103	0.00251	0.00102	0.0052
Fuzzy function	T_f	0.05665	0.143	0.00561	-

Table 5. Comparative results of user computational costs.

Protocols	User	D1 (ms)	D2 (ms)	D3 (ms)
Amin et al. [13]	$9T_h$	0.0093	0.0226	0.0092
Martinez et al. [17]	$11T_h + T_{de}$	0.0526	0.2275	0.2112
Zhou et al. [18]	$10T_h$	0.0103	0.0251	0.0102
Kang et al. [19]	$8T_h$	0.0082	0.0201	0.0082
Ours	$T_f + 10T_h$	0.0697	0.1681	0.0158

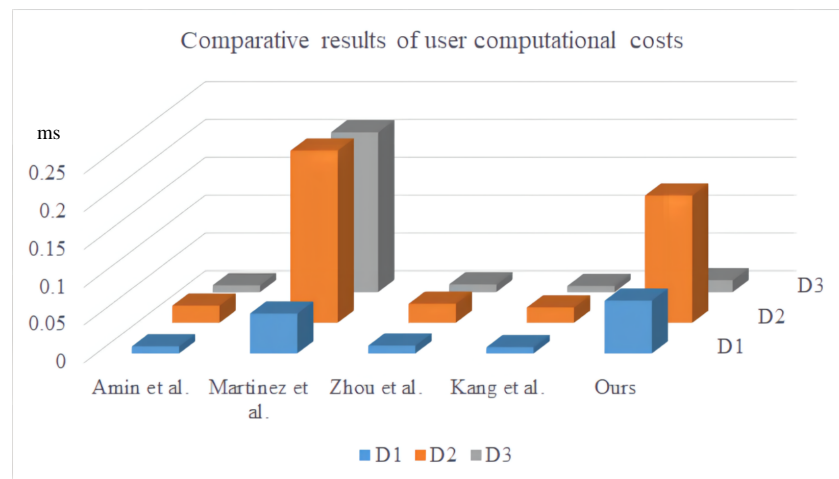


Figure 9. Comparative results of user computational costs [13,17–19].

We used a computer with a windows 10 operating system, Intel (R) core (TM) i5-8500CPU@ 3.00GHz/3.00G processor, and 8 GB memory, to simulate the server’s computational costs. IntelliJ idea software version 2019.3 was used for development. It is based on the Java pairing library, signature library, and symmetric encryption/decryption function. Various operations were run 10 times on this computer, and the average value was used as the reference time. Table 4 lists the results. Table 6 shows the comparison results. As shown in Table 6, the time required for our proposed protocol was only four hashes more than Kang et al.’s [19] and Amin et al.’s [13] protocols, that is, 0.0208 ms.

Table 6. Comparative results of server computational costs.

Protocols	Cloud Server	Control Server	Total (ms)
Amin et al. [13]	$4T_h$	$10T_h$	0.0728
Martinez et al. [17]	$6T_h + 2T_{de} + T_{en}$	$34T_h + 2T_{en}$	14.5774
Zhou et al. [18]	$7T_h$	$20T_h$	0.1404
Kang et al. [19]	$3T_h$	$11T_h$	0.0728
Ours	$5T_h$	$13T_h$	0.0936

To calculate the communication cost, we set the length of one-way hash H as 256 bits, timestamp T to 32 bits, string s to 160 bits, identity ID to 160 bits, random number Z_p^* to 160 bits, and encryption operation E to 256 bits. In addition, assume that the fuzzy extractor needs to store 8 bits. From this definition, the protocol of Zhou et al. [18] can be concluded to require $3|ID| + 10|s| + 6|H| + 3|T|$, that of Amin et al. [13] requires $3|ID| + 8|s| + 6|H| + 3|T|$, that of Martinez et al. [17] requires $3|ID| + 18|s| + |H| + 2|E| + 2|Z_p^*|$, that of Kang et al. [19] requires $3|ID| + 10|s| + 6|H| + 3|T|$, and our protocol requires $3|ID| + 15|s| + 5|H| + 3|T|$. Table 7 and Figure 10 present the detailed comparison results. Evidently, our protocol has a lower communication cost than that of Martinez et al. [17], although the communication cost is higher than that of the other three protocols. Our protocol also provides higher security; their protocols have been proven to have various security problems. Therefore, our proposed protocol is secure, has a relatively good performance, and is suitable for cloud computing environments.

For the storage costs, we consider the parameters required to store each entity in each entity in the registration phase. The number of numbers required for various parameters is the same as discussed above. The comparison results are shown in Figure 11. It can be seen from the figure that our storage cost is slightly higher than the protocols of Amin et al. [13] and Kang et al. [19].

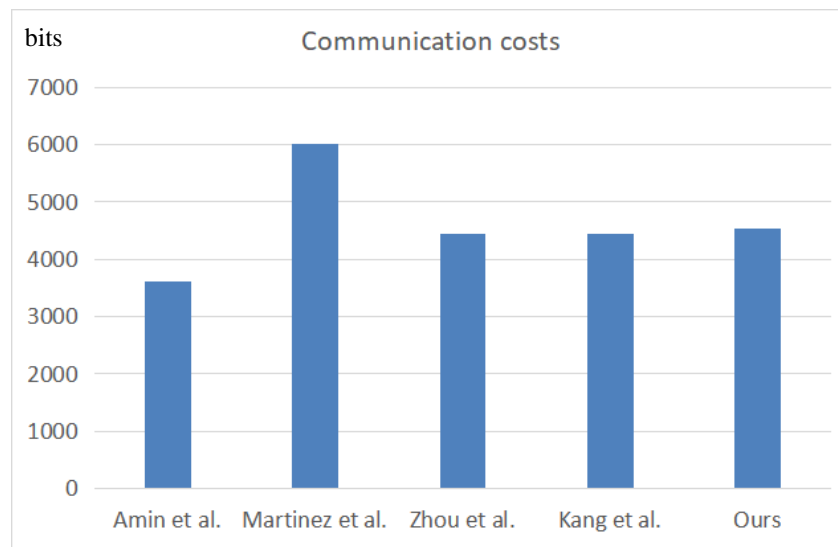


Figure 10. Comparative results of communication costs [13,17–19].

Table 7. Comparisons in terms of communication and storage costs.

Protocols	Number of Rounds	Communication Costs (Bits)	Storage Costs (Bits)	Security
Amin et al. [13]	5	3680	1152	Insecure
Martinez et al. [17]	6	6016	1664	Insecure
Zhou et al. [18]	4	4448	2112	Insecure
Kang et al. [19]	2	4000	1278	Cannot resist offline password guessing attack
Ours	5	4544	1320	Provable security

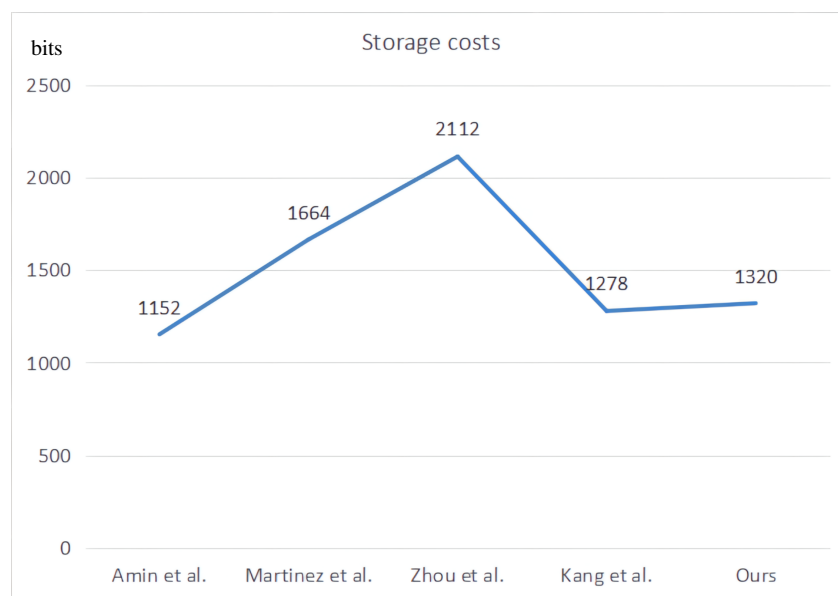


Figure 11. Comparative results of storage costs [13,17–19].

In terms of energy costs, due to the strong computing power and good performance of the server, we do not consider its energy cost. We use “ampere” APP to measure the current and voltage of each mobile phone the three devices during operation, and the results are shown in the Table 8. According to the formula $W = U \cdot I \cdot t$, the power consumption required by each device was calculated. The results are shown in Table 9 and Figure 12. The energy

costs are different for different devices. It can be seen from the figure that, although our protocol is not the best, it is better than that of Martinez et al. [17] and our protocol is secure.

Table 8. Voltage and current of devices.

Devices	U (V)	I (mA)
D1	4.08	531
D2	610	3.58
D3	508	4.08

Table 9. Energy costs.

Protocols	D1 (uJ)	D2 (uJ)	D3 (uJ)
Amin et al. [13]	20.148	49.354	19.068
Martinez et al. [17]	113.957	496.814	437.74
Zhou et al. [18]	22.315	54.813	21.14
Kang et al. [19]	17.751	43.894	16.996
Ours	151.004	367.097	32.748

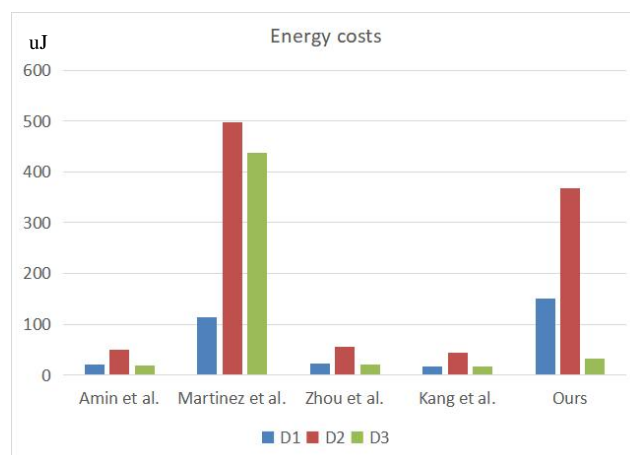


Figure 12. Comparative results of energy costs [13,17–19].

6. Conclusions and Discussion

IoT-enabled cloud computing environment is an open environment, and its main threat is data leakage. Because a large number of customers' privacy data are stored on the cloud server, once the data is leaked, it will lead to not only the disclosure of trade secrets, intellectual property rights, and personal privacy but also a devastating blow to the enterprise. In addition, the communication information of various entities is transmitted on the open channel. Attackers can launch attacks by intercepting the information on the open channel. Moreover, from Table 1, we can know that most of the existing schemes have been attacked, such as man in the middle attack, simulation attack, etc.

In this paper, we propose a protocol to solve the security problem in this environment. To verify the security, an informal security analysis was conducted, and the ProVerif and ROR models were adopted for formal security analysis. Finally, the protocol's security and performance were measured against those of other protocols. The proposed protocol can be concluded to satisfy basic security requirements. Therefore, our protocol is more suitable for this environment.

Author Contributions: Conceptualization, T.-Y.W.; methodology, T.-Y.W. and Q.M.; software, S.K.; formal analysis, P.Z.; investigation, T.-Y.W.; writing—original draft preparation, T.-Y.W., Q.M., S.K. and P.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research study received no external funding.

Data Availability Statement: The data are contained within the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
ROR	real-oracle random
AKA	authentication and key agreement
DoS	denial of service

References

1. Goudos, S.K.; Dallas, P.I.; Chatziefthymiou, S.; Kyriazakos, S. A survey of IoT key enabling and future technologies: 5G, mobile IoT, semantic web and applications. *Wirel. Pers. Commun.* **2017**, *97*, 1645–1675. [\[CrossRef\]](#)
2. Huang, X.; Xiong, H.; Chen, J.; Yang, M. Efficient Revocable Storage Attribute-based Encryption with Arithmetic Span Programs in Cloud-assisted Internet of Things. *IEEE Trans. Cloud Comput.* **2021**. [\[CrossRef\]](#)
3. Xiong, H.; Chen, J.; Mei, Q.; Zhao, Y. Conditional privacy-preserving authentication protocol with dynamic membership updating for VANETs. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 2089–2104. [\[CrossRef\]](#)
4. Wu, T.Y.; Wang, T.; Lee, Y.Q.; Zheng, W.; Kumari, S.; Kumar, S. Improved authenticated key agreement scheme for fog-driven IoT healthcare system. *Secur. Commun. Netw.* **2021**, *2021*, 6658041. [\[CrossRef\]](#)
5. Meng, Z.; Pan, J.S.; Tseng, K.K. PaDE: An enhanced Differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization. *Knowl. Based Syst.* **2019**, *168*, 80–99. [\[CrossRef\]](#)
6. Xue, X.; Zhang, J. Matching large-scale biomedical ontologies with central concept based partitioning algorithm and adaptive compact evolutionary algorithm. *Appl. Soft Comput.* **2021**, *106*, 107343. [\[CrossRef\]](#)
7. Pan, J.S.; Liu, N.; Chu, S.C.; Lai, T. An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems. *Inf. Sci.* **2021**, *561*, 304–325. [\[CrossRef\]](#)
8. Chandra, S.; Yafeng, W. Cloud things construction—The integration of Internet of Things and cloud computing. *Future Gener. Comput. Syst.* **2016**, *56*, 684–700.
9. Díaz, M.; Martín, C.; Rubio, B. State-of-the-art, challenges, and open issues in the integration of Internet of Things and cloud computing. *J. Netw. Comput. Appl.* **2016**, *67*, 99–117. [\[CrossRef\]](#)
10. Sun, P. Security and privacy protection in cloud computing: Discussions and challenges. *J. Netw. Comput. Appl.* **2020**, *160*, 102642. [\[CrossRef\]](#)
11. Rashid, A.; Chaturvedi, A. Cloud computing characteristics and services: A brief review. *Int. J. Comput. Sci. Eng.* **2019**, *7*, 421–426. [\[CrossRef\]](#)
12. Odelu, V.; Das, A.K.; Kumari, S.; Huang, X.; Wazid, M. Provably secure authenticated key agreement scheme for distributed mobile cloud computing services. *Future Gener. Comput. Syst.* **2017**, *68*, 74–88. [\[CrossRef\]](#)
13. Amin, R.; Kumar, N.; Biswas, G.; 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. [\[CrossRef\]](#)
14. Wu, F.; Li, X.; Xu, L.; Sangaiyah, A.K.; Rodrigues, J.J. Authentication protocol for distributed cloud computing: An explanation of the security situations for Internet-of-Things-enabled devices. *IEEE Consum. Electron. Mag.* **2018**, *7*, 38–44. [\[CrossRef\]](#)
15. Wang, C.; Ding, K.; Li, B.; Zhao, Y.; Xu, G.; Guo, Y.; Wang, P. An enhanced user authentication protocol based on elliptic curve cryptosystem in cloud computing environment. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 3048697. [\[CrossRef\]](#)
16. Pan, J.S.; Sun, X.X.; Chu, S.C.; Abraham, A.; Yan, B. Digital watermarking with improved SMS applied for QR code. *Eng. Appl. Artif. Intell.* **2021**, *97*, 104049. [\[CrossRef\]](#)
17. Martínez-Peláez, R.; Toral-Cruz, H.; Parra-Michel, J.R.; García, V.; Mena, L.J.; Félix, V.G.; Ochoa-Brust, A. An enhanced lightweight IoT-based authentication scheme in cloud computing circumstances. *Sensors* **2019**, *19*, 2098. [\[CrossRef\]](#)
18. 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. [\[CrossRef\]](#)
19. Kang, B.; Han, Y.; Qian, K.; Du, J. Analysis and improvement on an authentication protocol for IoT-enabled devices in distributed cloud computing environment. *Math. Probl. Eng.* **2020**, *2020*, 1970798. [\[CrossRef\]](#)
20. Luo, Y.; Zheng, W.; Chen, Y.C. An anonymous authentication and key exchange protocol in smart grid. *J. Netw. Intell.* **2021**, *6*, 206–215.
21. Wu, T.Y.; Yang, L.; Luo, J.N.; Ming-Tai Wu, J. A Provably Secure Authentication and Key Agreement Protocol in Cloud-Based Smart Healthcare Environments. *Secur. Commun. Netw.* **2021**, *2021*, 2299632. [\[CrossRef\]](#)
22. 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. [\[CrossRef\]](#)
23. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Conti, M.; Jo, M. Design of secure user authenticated key management protocol for generic IoT networks. *IEEE Internet Things J.* **2017**, *5*, 269–282. [\[CrossRef\]](#)

24. Wu, F.; Li, X.; Xu, L.; Vijayakumar, P.; Kumar, N. A novel three-factor authentication protocol for wireless sensor networks with IoT notion. *IEEE Syst. J.* **2020**, *15*, 1120–1129. [[CrossRef](#)]
25. Tsai, J.L.; Lo, N.W. A privacy-aware authentication scheme for distributed mobile cloud computing services. *IEEE Syst. J.* **2015**, *9*, 805–815. [[CrossRef](#)]
26. Irshad, A.; Sher, M.; Ahmad, H.F.; Alzahrani, B.A.; Chaudhry, S.A.; Kumar, R. An improved multi-server authentication scheme for distributed mobile cloud computing services. *KSII Trans. Internet Inf. Syst. (TIIS)* **2016**, *10*, 5529–5552.
27. Sadri, M.J.; Asaar, M.R. An anonymous two-factor authentication protocol for IoT-based applications. *Comput. Netw.* **2021**, *199*, 108460. [[CrossRef](#)]
28. He, D.; Kumar, N.; Khan, M.K.; Wang, L.; Shen, J. Efficient privacy-aware authentication scheme for mobile cloud computing services. *IEEE Syst. J.* **2016**, *12*, 1621–1631. [[CrossRef](#)]
29. Xiong, L.; Peng, D.; Peng, T.; Liang, H. An enhanced privacy-aware authentication scheme for distributed mobile cloud computing services. *KSII Trans. Internet Inf. Syst. (TIIS)* **2017**, *11*, 6169–6187.
30. 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.* **2020**, *108*, 1267–1286. [[CrossRef](#)]
31. Yu, S.; Park, K.; Park, Y. A secure lightweight three-factor authentication scheme for IoT in cloud computing environment. *Sensors* **2019**, *19*, 3598. [[CrossRef](#)] [[PubMed](#)]
32. Wang, F.; Xu, G.; Xu, G.; Wang, Y.; Peng, J. A robust IoT-based three-factor authentication scheme for cloud computing resistant to session key exposure. *Wirel. Commun. Mob. Comput.* **2020**, *2020*, 3805058. [[CrossRef](#)]
33. Huang, H.; Lu, S.; Wu, Z.; Wei, Q. An efficient authentication and key agreement protocol for IoT-enabled devices in distributed cloud computing architecture. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 1–21. [[CrossRef](#)]
34. Li, N.; Guo, F.; Mu, Y.; Susilo, W.; Nepal, S. Fuzzy extractors for biometric identification. In Proceedings of the IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 667–677.
35. Canetti, R.; Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory And Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2045, pp. 453–474.
36. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [[CrossRef](#)]
37. Canetti, R.; Goldreich, O.; Halevi, S. The random oracle methodology, revisited. *J. ACM* **2004**, *51*, 557–594. [[CrossRef](#)]
38. Odelu, V.; Das, A.K.; Goswami, A. A secure biometrics-based multi-server authentication protocol using smart cards. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 1953–1966. [[CrossRef](#)]
39. Wang, D.; Cheng, H.; Wang, P.; Huang, X.; Jian, G. Zipf’s law in passwords. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2776–2791. [[CrossRef](#)]
40. Blanchet, B. A computationally sound mechanized prover for security protocols. *IEEE Trans. Dependable Secur. Comput.* **2008**, *5*, 193–207. [[CrossRef](#)]
41. Abadi, M.; Fournet, C. Mobile values, new names, and secure communication. *ACM Sigplan Not.* **2001**, *36*, 104–115. [[CrossRef](#)]