Rotating Behind Security: An enhanced authentication protocol for IoT-enabled devides in distributed cloud computing architecture

Tsu-Yang Wu
Shandong University of Science and Technology
Fangfang Kong
Shandong University of Science and Technology
Qian Meng
Shandong University of Finance and Economics
Saru Kumari
Shandong University of Science and Technology
Chien-Ming Chen (chienmingchen@ieee.org)
Shandong University of Science and Technology https://orcid.org/0000-0002-6502-472X

Research Article

Keywords: IoT, cloud computing, enhanced authentication protocol

Posted Date: April 25th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1554621/v1

RESEARCH

Rotating behind Security: An enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture

Tsu-Yang Wu[1], Fangfang Kong[1], Qian Meng[1], Saru Kumari[2] and Chien-Ming Chen[1*]

*Correspondence:
chienmingchen@ieee.org
[1]College of Computer Science and Engineering, Shandong University of Science and Technology, 266590, Qingdao, China
Full list of author information is available at the end of the article

Abstract

As Internet of Things (IoT) technology continues to advance, IoT devices have permeated every part of life. The emergence of cloud computing has become a key technology for solving the memory limitation problem of IoT devices, greatly promoting resource sharing, facilitating user access to information at any time, and providing IoT services. However, IoT message is transferred over public channels and is therefore accessible to attacks in the IoT-based cloud computing environment, resulting in user and server privacy leakage. To solve this issue, Amin et al. proposed an authentication protocol for use in this environment. However, Kang et al. proved that their protocol had many security vulnerabilities and proposed an improved authentication protocol. Recently, Huang et al. found security vulnerabilities in Kang et al.'s protocol and designed an enhanced protocol of their own. In this article, we first demonstrate that Huang et al.'s enhanced protocol is vulnerable to privileged insider attacks and temporary value leakage attacks. Then, based on their protocol an enhanced authentication protocol is proposed for IoT-enabled devices in a distributed cloud computing architecture. Finally, security analysis and performance comparisons are conducted, showing that our protocol has both greater security and better performance.

Keywords: IoT; cloud computing; enhanced authentication protocol

1 Introduction

The Internet of Things (IoT) collects information from sensor devices shared between different devices [1]. With the development of Internet and artificial intelligence technology [2–4], IoT technology is gradually being applied to various fields, and plays a great role in areas such as the Internet of vehicles [5, 6] and smart grids [7, 8]. IoT devices have penetrated every aspect of our lives [9, 10]. However, IoT devices have limited memory and are data-intensive [11], and some of them retain sensitive information about their users. As a result, protecting data in IoT devices has become a very important issue.

With the progress of Internet technology, cloud computing has emerged as new computer technology to provide computing and storage services. The ability to process large quantities of heterogeneous data and devices efficiently has made cloud computing a key technology [12] that provides on-demand access to storage and computing resources [13]. Private clouds are deployed within businesses to fulfill their needs and provide secure control over data security and service quality. Cloud

computing has become pervasive, so attention must be paid to information and data security.

Information travels over open and insecure channels between IoT devices in cloud computing environments. When users send queries to cloud servers, they are vulnerable data privacy leakage. Therefore, to protect the security of user request information, users need to complete identity authentication and establish session keys when sending requests to the cloud server. To solve this problem, QR code [14] technology was developed. Many authentication protocols have been proposed for IoT devices in the cloud computing environment. However, designing a secure user authentication protocol remains a challenge.

Turkanovic et al. [15] proposed an authentication and key agreement (AKA) protocol based on the concept of the IoT. However, Farash et al.'s [16] cryptanalysis proved that Turkanovic et al.'s [15] protocol was vulnerable to sensor node simulation attacks and smart card theft attacks, and could not guarantee the anonymity of users and sensor nodes.Then, Farash et al. [16] proposed a scheme to enhance security. Unfortunately, their scheme was proven to have many security vulnerabilities by Amin et al. [17]. Braeken et al. [18] proposed an authentication protocol using physically unclonable functions in IoT environments. Panda et al. [19] proposed an AKA protocol that uses elliptic curve cryptography (ECC) encryption. Challa et al. [20] proposed an improved signature-based AKA protocol for the IoT environment. Liu et al. [21] proposed a privacy protection authentication protocol based on shared authorization in the cloud computing environment. Subsequently, Kalra et al. [22] proposed a secure AKA protocol for the cloud server environment for the IoT, also using ECC encryption. Amin et al. [23] pointed out security vulnerabilities in the protocols designed by Xue et al. [24] and Chuang et al. [25]. Amin et al. [23] also designed a distributed cloud environment architecture and an AKA protocol based on a smart card. Unfortunately, Wang et al. [26] pointed out that Amin et al.'s [23] protocol does not provide forward confidentiality and cannot withstand offline dictionary attacks. Wang et al. [26] then designed a secure AKA protocol using ECC. Irshad et al. [27] designed a multi-server architecture AKA protocol using chaotic projection. Wu et al. [28] developed an AKA protocol for securing IoT devices with the help of cloud servers. Fan et al. [29] designed an AKA protocol using radio frequency identification (RFID) technology for the cloud computing environment based on the IoT. Yu et al. [30] confirmed the security vulnerabilities in He et al.'s [31] protocol and designed a new AKA protocol. Kumari et al. [32] designed an ECC-based AKA scheme. Wazid et al. [33] proposed a lightweight AKA protocol. Irshad et al. [34] designed a new type of lightweight unpaired AKA protocol. Rangwani et al. [35] designed a secure AKA protocol using ECC. Zargar et al. [36] designed a lightweight AKA protocol in an IoT-based cloud environment.

Kang et al. [37] conducted cryptanalysis of Amin et al.'s [23] protocol, proved that their protocol suffered from security vulnerabilities, and proposed an improved AKA protocol. In 2021, Huang et al. [38] demonstrated that the protocol of Kang et al. [37] was vulnerable to offline password attacks, and improved the protocol using lightweight hash functions. We conducted cryptanalysis on the protocol of Huang et al. [38] and proved that it has two security defects. Specifically, it is unable to resist either privileged insider attacks or temporary value leakage attacks. Therefore, we

Table 1: Notations

Notations	Meanings
TCS	The control server
S_j	i^{th} cloud server
$CSID_j$	S_j'sIdentity
U_i	i^{th} user
UID_i	U_i's Identity
UPW_i	U_i's Password
BIO_i	U_i's Biometric
x	TCS's private secret key

designed an enhanced authentication protocol for IoT-enabled devices in distributed cloud computing architecture. We demonstrated that the proposed AKA protocol has good security by formal and informal security analysis to validate the protocol's security. Our protocol can withstand known attacks, according to security analysis. The results of the performance evaluation suggest that our protocol is feasible and can be implemented at a low computational cost.

The structure of this article is as follows: In Section 2, we briefly outline the methods and experimental of this article. In Section 3 , we outline the protocol of Huang et al. [38], and point out the security issues therein. The proposed protocol is described in detail in Section 4, results and discussion in Section 5. Finally, summarized in Section 6.

2 Methods and experimental

We employ the same architectural model as Huang et al. [38] in this article. There are three entities in the architecture model: User U_i, cloud provider server S_j, and control server TCS. U_i can obtain cloud server services by using IoT devices. S_j provides computing and storage services. TCS is in charge of U_i and S_j registration and authentication. First, U_i and S_j register with TCS. U_i and S_j are authenticated by TCS, which will occur when U_i wishes to retrieve data from the server. Session key is generated with the help of TCS. Our protocol performs only two kinds of computing operations: hash functions and XOR operations.

We use real-oracle-random (ROR) model for formal analysis, which proves that our protocol is secure. Our protocol security was verified using the ProVerif tool on Lenovo desktop with Windows 10 operating system. Verify protocol security through informal security analysis. The results of the performance evaluation suggest that our protocol is feasible and can be implemented at a low computational cost.

3 Security Analysis of Huang el at.'s Protocol

This section we review Huang et al's [38] protocol and point out security issues in the protocol. User U_i, cloud provider server S_j, and control server TCS are the three entities that make up the protocol. The protocol consists of three stages: registration phase, login phase, authentication and key agreement phase. Table 1 shows the symbols and instructions used in this article.

3.1 Review Huang el at.'s Protocol

3.1.1 Registration Phase

Registration of U_i and S_j are performed separately, as follows.

S_j Registration Phase

(1) First, S_j chooses its identity $CSID_j$ and a random number R_j, then using the secure channel it sends $\{CSID_j, R_j\}$ to TCS.
(2) After receiving $\{CSID_j, R_j\}$ sent by S_j, TCS computes $TSID_j = h(CSID_j \parallel R_j)$, $RSID_j = h(TSID_j \parallel CSID_j \parallel y)$, where y is the private key that authenticates all S_j. Finally, TCS transmits $\{RSID_j\}$ to S_j over the secure channel.
(3) After receiving $\{RSID_j\}$ sent by TCS, S_j stores secret parameter $\{RSID_j, R_j\}$ in memory.

U_i Registration Phase

(1) U_i selects its identity UID_i, password UPW_i, biological characteristic BIO_i, and random number R_i. Then, U_i computes $RID_i = h(UID_i \parallel R_i)$, $UA_i = UPW_i \oplus h(BIO_i)$, and $UC_i = R_i \oplus UA_i$. Next, U_i sends $\{UID_i, RID_i, UA_i\}$ to the TCS.
(2) After receiving $\{UID_i, RID_i, UA_i\}$ sent by U_i, TCS first validates the identity of U_i. If UID_i is not registered, TCS calculates $TD_i = h(UID_i \parallel UA_i)$, $TE_i = h(RID_i \parallel x)$, and $TF_i = TE_i \oplus UA_i$. Then, TCS stores the data $\{TD_i, TF_i, h(.)\}$ in smart card (SC) and sends SC to U_i.
(3) U_i receives SC and stores UC_i to SC. Finally, SC records the information $\{UC_i, TD_i, TF_i, h(.)\}$

3.2 Login Phase

U_i puts SC into the IoT-enabled device, then enters UID_i, UPW_i, and BIO_i. The SC computes $UA_i = UPW_i \oplus h(BIO_i)$, $TD_i^* = h(UID_i \parallel UA_i)$, and then performs authentication by checking $TD_i^* \stackrel{?}{=} TD_i$. If authentication is successful, U_i logs in successfully.

3.3 Authentication and Key Agreement Phase

Mutual authentication is performed between U_i, S_j, and TCS when U_i signs in. The specifics of this phase are as follows.

(1) U_i generates random UN_i and timestamp TS_i, and chooses $CSID_j$. Then, U_i computes $R_i = UC_i \oplus UA_i$, $RID_i = h(UID_i \parallel R_i)$, $TE_i = TF_i \oplus UA_i$, $UG_i = h(RID_i \parallel CSID_j \parallel UN_i \parallel TS_i \parallel TE_i)$, $UH_i = TE_i \oplus UN_i$, and $UJ_i = CSID_j \oplus h(TE_i \parallel UN_i)$. After that, U_i sends message $M_1 = \{UG_i, UH_i, UJ_i, RID_i, TS_i\}$ to S_j
(2) After receiving M_1, S_j verifies that timestamp $|TS_j - TS_i| \leqq \Delta T$. If the timestamp is valid, S_j generates random CN_j and timestamp TS_j. S_j then computes $CK_j = RSID_j \oplus CN_j$, $CL_j = h(CN_j \parallel RSID_j \parallel RID_i \parallel UG_i \parallel TS_j)$, and sends message $M_2 = \{CK_j, CL_j, TSID_j, UG_i, UH_i, UJ_i, RID_i, TS_i, TS_j\}$ to TCS.
(3) After receiving M_2, TCS verifies that timestamp $|TS_{SC} - TS_j| \leqq \Delta T$. If the timestamp is not fresh, the session is terminated. Otherwise, TCS computes $TE_i = h(RID_i \parallel x)$, $UN_i = TF_i \oplus TD_i$, $CSID_j = UJ_i \oplus h(TE_i \parallel UN_i)$, and $UG_i^* = h(RID_i \parallel CSID_j \parallel UN_i \parallel TS_i \parallel TE_i)$. TCS checks $UG_i^* \stackrel{?}{=} UG_i$ to verify the identity of U_i. If the identification is successful, TCS computes

$RSID_j = h(TSID_j \parallel CSID_j \parallel y)$, $CN_j = RSID_j \oplus CK_j$, and $CL_j^* = h(CN_j \parallel RSID_j \parallel RID_i \parallel UG_i \parallel TS_j)$. TCS checks $CL_j^* \stackrel{?}{=} CL_j$ to verify the identity of S_j. If the verification succeeds, TCS chooses TN_{CS}. Then, TCS computes $TP_{CS} = CN_j \oplus TN_{CS} \oplus h(UN_i \parallel TE_i \parallel UH_i)$, $TR_{CS} = UN_i \oplus TN_{CS} \oplus h(RSID_j \parallel CN_j)$, $SK_{CS} = h(UN_i \oplus CN_j \oplus TN_{CS})$, $TQ_{CS} = h((UN_i \oplus TN_{CS}) \parallel SK_{CS})$, $TV_{CS} = h((UN_i \oplus TN_{CS}) \parallel SK_{CS})$. Finally, TCS sends message $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$ to S_j.

(4) After receiving M_3, S_j computes $CW_j = h(RSID_j \parallel CN_j)$, $UN_i \oplus TN_{CS} = TR_{CS} \oplus CW_j$, $SK_j = h(UN_i \oplus TN_{CS} \oplus CN_j)$, and $TV_{CS}^* = h((UN_i \oplus TN_{CS}) \parallel SK_j)$. S_j verifies whether the identity of TCS is valid by checking $TV_{CS}^* \stackrel{?}{=} TV_{CS}$. If true, S_j transmits message $M_4 = \{TP_{CS}, TQ_{CS}\}$ to U_i.

(5) After receiving M_4, U_i computes $UZ_i = h(UN_i \parallel TE_i \parallel UH_i)$, $CN_j \oplus TN_{CS} = TP_{CS} \oplus UZ_i$, $SK_i = h(CN_j \oplus TN_{CS} \oplus UN_i)$, and $TQ_{CS}^* = h((CN_j \oplus TN_{CS}) \parallel SK_i)$. U_i verifies whether the identities of TCS and S_j are valid by checking $TQ_{CS}^* \stackrel{?}{=} TQ_{CS}$. If true, the three participants U_i, S_j, and TCS negotiate a session key $SK = h(UN_i \oplus CN_j \oplus TN_{CS})$.

3.4 Cryptanalysis of Huang et al.'s Protocol

We discovered that Huang et al.'s [38] protocol is unable to resist either privileged insider attacks or temporary value leakage disclosure attacks.

3.4.1 Adversary Model

We describe the capabilities of the attacker (A) in detail below:

(1) A can eavesdrop on, update, or capture information during the transmission of data through a public channel.

(2) A can obtain information stored in the cloud server.

(3) A can intercept private values in user memory or cloud server memory.

3.4.2 Privileged Insider Attacks

Assuming that A obtains the value $\{RSID_j, R_i\}$ stored in S_j, SK is calculated as follows:

(1) First, A captures CK_i, which is transmitted over the public channel.

(2) A obtains the value of CN_j by calculating $CN_j = CK_i \oplus RSID_j$. Then, A calculates $CW_j = h(RSID_j \parallel CN_j)$ and $UN_i \oplus TN_{CS} = TR_{CS} \oplus CW_j$.

(3) A can calculate $SK = h(UN_i \oplus TN_{CS} \oplus CN_j)$.

Therefore, Huang et al.'s [38] protocol is unable to resist privileged insider attacks.

3.4.3 Temporary Value Disclosure Attacks

We assume that A can obtain temporary values from U_i or S_j.

Case one: A obtains the temporary value UN_i of U_i. The following steps can be used to calculate SK:

(1) A first intercepts $\{UH_i, TP_{CS}\}$, which is transmitted over the public channel.

(2) A obtains the value of TE_i by computing $TE_i = UH_i \oplus UN_i$. Then A calculates $UZ_i = h(UN_i \parallel TE_i \parallel UH_i)$ and $CN_j \oplus TN_{CS} = TP_{CS} \oplus UZ_i$.

(3) A can calculate $SK_i = h(CN_j \oplus TN_{CS} \oplus UN_i)$.

Case two: A obtains the temporary value CN_j of S_j. The following steps can be used to calculate SK:

(1) A first intercepts $\{CK_j, TR_{CS}\}$, which is sent over the public channel.
(2) A calculates $RSID_j = CK_j \oplus CN_j$ to obtain the value of $RSID_j$. Then A calculates $CW_j = h(RSID_j \parallel CN_j)$ and $UN_i \oplus TN_{CS} = TR_{CS} \oplus CW_j$.
(3) A can calculate $SK_j = h(UN_i \oplus TN_{CS} \oplus CN_j)$.

In both cases, Huang et al.'s [38] protocol is therefore unable to resist temporary value disclosure attacks.

4 Proposed Protocol

In this section, we describe an enhanced authentication protocol for IoT-enabled devices in a distributed cloud computing environment. The protocol consists of three phases: S_j registration, U_i registration, and login authentication.

4.1 S_j Registration Phase

In this phase, data is transmitted over a secure channel. The S_j registration phase proceeds as illustrated in Figure 1:

(1) S_j chooses its identity $CSID_j$ and a random number R_j. Next, S_j calculates $RSID_j = h(CSID_j \parallel R_j)$. Finally, S_j sends $\{RSID_j, CSID_j\}$ to TCS.
(2) After TCS receives $\{RSID_j, CSID_j\}$, it computes $TSID_j = h(RSID_j \parallel CSID_j \parallel x)$ and sends the result to S_j .
(3) After S_j receives $\{TSID_j\}$, it computes $CZ_j = CSID_j \oplus TSID_j$. Finally, S_j stores $\{CZ_j\}$ in its database.

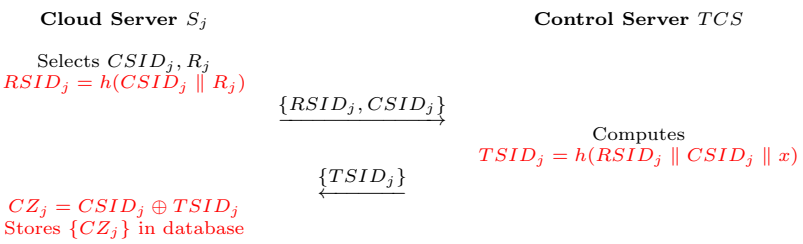

Figure 1: S_j Registration Phase

4.2 U_i Registration Phase

The U_i registration phase proceeds as illustrated in Figure 2:

(1) U_i chooses its own identity UID_i, password UPW_i, biological characteristic BIO_i, and random number R_i. Then U_i computes $RID_i = h(UID_i \parallel R_i)$ and sends $\{UID_i, RID_i\}$ to TCS.
(2) After TCS receives $\{UID_i, RID_i\}$ from U_i, TCS verifies UID_i. If UID_i is a registered identity, TCS rejects the registration request from U_i. Otherwise, TCS chooses random number t_i and calculates $TB_i = h(RID_i \parallel x \parallel t_i)$. Then, TCS stores the data $\{RID_i, t_i\}$ in its database. Finally, TCS sends $\{TB_i, t_i\}$ to U_i.
(3) U_i receives $\{TB_i, t_i\}$ from TCS, computes $UA_i = UPW_i \oplus h(BIO_i)$, $UC_i = R_i \oplus UA_i, UE_i = h(RID_i \parallel UA_i), UF_i = TB_i \oplus UA_i$, and $UG_i = t_i \oplus h(UID_i \parallel UPW_i \parallel R_i)$. Finally, U_i stores $\{UC_i, UE_i, UF_i, UG_i, h(.)\}$ in SC.

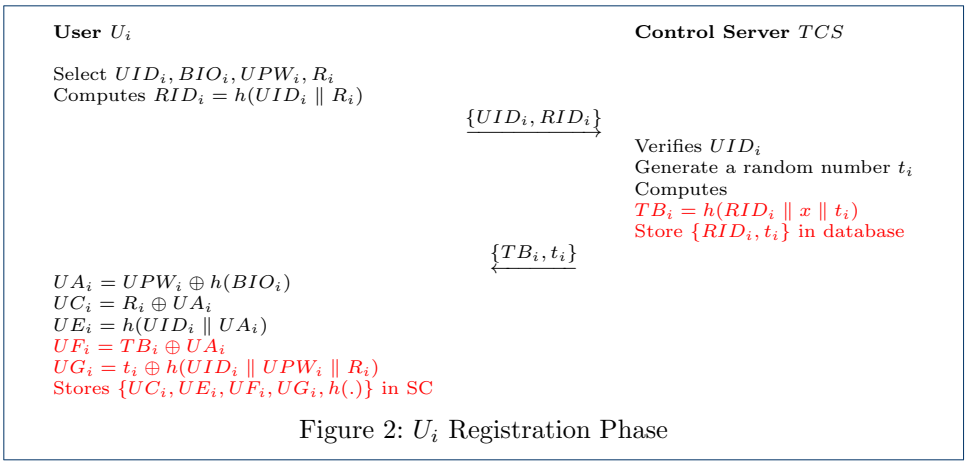

Figure 2: U_i Registration Phase

4.3 Login Authentication Phase

In this phase, U_i and S_j are authenticated by TCS, which will occur when U_i wishes to retrieve data from the server. Session key SK incorporating U_i, S_j and TCS is generated with the help of TCS. The login authentication phase proceeds as illustrated in the Figure 3:

(1) U_i inputs UID_i, UPW_i, and BIO_i, and computes $UA_i = UPW_i \oplus h(BIO_i)$, $UE_i^* = h(UID_i \| UA_i)$. U_i checks whether $UE_i^* \stackrel{?}{=} UE_i$. If they are equal, U_i chooses random number UN_i, timestamp TS_i, and $CSID_j$, which is the identity of S_j. U_i then computes $R_i = UC_i \oplus UA_i$, $RID_i = h(UID_i \| R_i)$, $TB_i = UF_i \oplus UA_i$, $UH_i = h(RID_i \| CSID_j \| UN_i \| TS_i \| TB_i)$, $t_i = h(UID_i \| UPW_i \| R_i) \oplus UG_i, UJ_i = h(TB_i \| t_i) \oplus UN_i$, and $UK_i = CSID_j \oplus h(TB_i \| UN_i \| t_i)$. Finally, U_i sends $M_1 = \{UH_i, UJ_i, UK_i, RID_i, TS_i\}$ to S_j.

(2) After receiving M_1 from U_i, S_j checks whether $|TS_j - TS_i| \leqq \Delta T$. If the timestamp is valid, S_j chooses random number CN_j. Then S_j computes $TSID_j = CZ_j \oplus CSID_j$, $CL_j = h(TSID_j) \oplus CN_j$, and $CM_j = h(CN_j \| CSID_j \| TSID_j \| RID_i \| UH_i \| TS_j)$. Finally, S_j sends $M_2 = \{UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_i, TS_j\}$ to TCS.

(3) After receiving the message from S_j, TCS checks whether $|TS_{CS} - TS_j| \leqq \Delta T$. If the timestamp is valid, TCS retrieves $\{t_i\}$ from its database using RID_i. TCS computes $TB_i = h(RID_i \| x \| t_i)$, $UN_i = UJ_i \oplus h(TB_i \| t_i)$, $CSID_j = UK_i \oplus h(TB_i \| UN_i \| t_i)$, and $UH_i^* = h(RID_i \| CSID_j \| UN_i \| TS_i \| TB_i)$. TCS verifies the identity of U_i by comparing $UH_i^* \stackrel{?}{=} UH_i$. If the authentication succeeds, it means that U_i is valid. Then TCS computes $TSID_j = h(RSID_j \| CSID_j \| x)$, $CN_j = h(TSID_j) \oplus CL_j$, and $CM_j^* = h(CN_j \| CSID_j \| TSID_j \| RID_i \| UH_i \| TS_j)$. TCS verifies the identity of S_j by comparing $CM_j^* \stackrel{?}{=} CM_j$. If the terms are equal, the identity of S_j is valid. Then TCS chooses random number N_{CS} and computes $TP_{CS} = CN_j \oplus TN_{CS} \oplus h(UN_i \| TB_i \| CSID_j)$, $TR_{CS} = UN_i \oplus TN_{CS} \oplus h(TSID_j \| CSID_j \| CN_j)$, $SK_{CS} = h(UN_i \oplus CN_j \oplus TN_{CS})$, $TQ_{CS} = h((CN_j \oplus TN_{CS}) \| SK_{CS})$, and $TV_{CS} = h((UN_i \oplus TN_{CS}) \| SK_{CS})$. Finally, TCS sends $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$ to S_j.

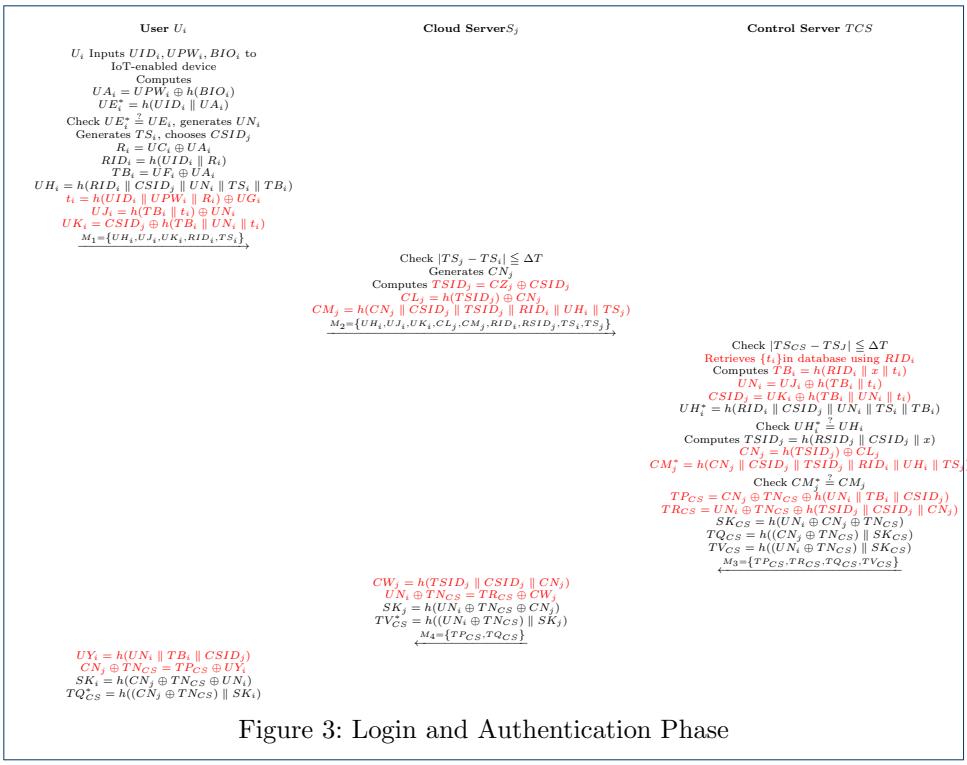

Figure 3: Login and Authentication Phase

(4) After receiving M_3, S_j computes $CW_j = h(TSID_j \| CSID_j \| CN_j)$, $UN_i \oplus TN_{CS} = TR_{CS} \oplus CW_j$, $SK_j = h(UN_i \oplus TN_{CS} \oplus CN_j)$, and $TV_{CS}^* = h((UN_i \oplus TN_{CS}) \| SK_j)$. If they are equal, S_j successfully authenticates TCS. S_j and TCS successfully authenticate each other. Finally, S_j sends $M_4 = \{TP_{CS}, TQ_{CS}\}$ to U_i.

(5) After receiving M_4 from S_j, U_i computes $UY_i = h(UN_i \| TB_i \| CSID_j)$, $CN_j \oplus TN_{CS} = TP_{CS} \oplus UY_i$, $SK_i = h(CN_j \oplus TN_{CS} \oplus UN_i)$, and $TQ_{CS}^* = h((CN_j \oplus TN_{CS}) \| SK_i)$. If they are not equal, the authentication fails. Otherwise, U_i successfully authenticates TCS and S_j. U_i, S_j and TCS successfully authenticate each other. Finally, U_i, S_j and TCS authenticate each other and negotiate SK.

5 Results and Discussion

5.1 Formal Security Analysis

Canetti et al. [39] proposed ROR model, which measures protocol security by evaluating the probability of cracking SK successfully across repeated rounds of various games. Using the ROR model, we compute that the probability of breaking our protocol is extremely low, indicating that our protocol is secure.

5.1.1 ROR Model

We propose three entities: user U_i, cloud server S_j, and control server TCS. Here we use $\Pi_{U_i}^x$, $\Pi_{S_j}^y$, and Π_{TCS}^z to represent the x-th user instance, the y-th cloud server instance, and the z-th control server instance, respectively. Assume that A can perform the following queries for $W = \{\Pi_{U_i}^x, \Pi_{S_j}^y, \Pi_{TCS}^z\}$.

Table 2: Simulation of Oracles

On a query Send($\Pi^x_{U_i}$,start), assuming that $\Pi^x_{U_i}$ is a regular state. We do the following: select $UN_i, TS_i, CSID_j$, and compute $R_i, RID_i, TB_i, UH_i, t_i, UJ_i, UK_i$. Then, the query is answered by $M_1 = \{UH_i, UJ_i, UK_i, RID_i, TS_i\}$.
On a query Send($\Pi^y_{S_j}, (UH_i, UJ_i, UK_i, RID_i, TS_i)$), assuming $\Pi^y_{S_j}$ is a regular state. We do the following: select CN_j, TS_j and compute $TSID_j, CL_j, CM_j$. Then, the query is answered by $M_2 = \{UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_i, TS_j\}$.
On a query Send($\Pi^z_{CS}, ((UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_i, TS_j))$, assuming Π^z_{CS} that is a normal state, next, compute $TB_i, UN_i, CSID_j, UH^*_i$, check UH^*_i. If equal, continue to calculate $TSID_j, CN_j, CM^*_j$, and check CM^*_j. If equal, select TN_{CS} and compute $TP_{CS}, TR_{CS}, SK_{CS}, TQ_{CS}, TV_{CS}$. Then, the query is answered by $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$.
On a query Send($\Pi^y_{S_j}, (TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS})$), assuming $\Pi^y_{S_j}$ is a regular state. We do the following: compute $CW_j, UN_i \oplus TN_{CS}$, SK_j, TV^*_{CS} and check TV^*_{CS}. If the verification does not equal, it will be terminated. Otherwise, the query is answered is by $M_4 = \{TP_{CS}, TQ_{CS}\}$.
On a query, assuming that $\Pi^x_{U_i}$ is a regular state. We do the following: compute $UY_i, CN_j \oplus TN_{CS}$, SK_i,TQ^*_{CS}, and check TQ^*_{CS}. If validation fails, it is terminated. Finally, $\Pi^x_{U_i}$ accepts and terminates.
On a query $Execute$ query, we use the simulation of $Send$ query to do the following operations: $(UH_i, UJ_i, UK_i, RID_i, TS_i) \longleftarrow Send(\Pi^x_{U_i}$, start), $(UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_i, TS_j) \longleftarrow Send(\Pi^y_{S_j}, (UH_i, UJ_i, UK_i, RID_i, TS_i))$,$(TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}) \longleftarrow Send(\Pi^z_{CS}, (UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_i, TS_j))$,$(TP_{CS}, TQ_{CS}) \longleftarrow Send(\Pi^y_{S_j}, TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}))$. This query is answered by $(UH_i, UJ_i, UK_i, RID_i, TS_i)$, $(UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_i, TS_j)$, $(TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS})$ and (TP_{CS}, TQ_{CS}).
For a $Hash(string)$ query, if a record $(string, s)$ appears in the query, return $s = hash(string)$. Otherwise, select an element s, add the record to the list, and return s.
For a $Corrupt(\Pi^x_{U_i})$, if $(\Pi^x_{U_i})$ is accepted, The query returns the parameters $\{UC_i, UE_i, UF_i, UG_i, h(.)\}$ in SC.
On a $Test$ query, flip coin C. If $C = 1$, return the correct SK; otherwise, return a random string of equal length to SK.

(1) $Execute(W)$: If A executes this query, it can eavesdrop on all messages transmitted between $\Pi^x_{U_i}$, $\Pi^y_{S_j}$, and Π^z_{TCS} over the public channel.
(2) $Send(W, M)$: If A performs this query, it can send message M to W and receive a response from W.
(3) $Hash(string)$: If A performs this query, it can acquire the hash value by entering a string.
(4) $Corrupt(W)$: If A executes this query, it will retrieve private parameters of an entity, such as a long-term key, user parameters stored on SC, or a temporary value.
(5) $Test(W)$: If A executes this query, a coin C is flipped. If $C = 1$, indicating that the coin is up, then A obtains the correct SK. If $C = 0$, indicating that the coin is down, then A obtains any string of the same length as SK.

5.1.2 ROR Proof

Theorem. In the ROR model, if A performs the queries $Execute(W)$, $Send(W, M)$, $Hash(string)$, $Corrupt(W)$, and $Test(W)$, the probability that A can break the proposed protocol P in polynomial time is: $Adv^{\mathcal{P}}_{\mathcal{A}}(\xi) \leq q_{send}/2^{l-2} + 3q^2_{hash}/2^l + 2max\{C' \cdot q^{s'}_{send}, q_{send}/2^l\}$, where q_{send} and q_{hash} indicate the times of the $Send$ and $Hash$ queries, respectively, l represents the number of bits of biometric information, and C' and s' are two constants.

Proof. In the ROR model, we have seven-game rounds represented as GM_i, $i = 0, 1...6$. $Succ_{\mathcal{A}}$ denotes the probability of winning numerous rounds in the game, and

$Adv_{\mathcal{A}}$ denotes the advantage of breaking the protocol. Table 2 shows the specific simulation of queries in real attacks. The following are the specific proof steps:

GM_0: GM_0 is the first round of the game and does not perform any queries. We start the round by flipping coin C. The probability of A successfully cracking P is:

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

GM_1: GM_1 performs one more $Execute(W)$ operation than GM_0. In this round, A intercepts only messages $\{M_1, M_2, M_3, M_4\}$ transmitted over the public channel. Since the values of UN_i, CN_j, and TN_{CS} are unknown, A cannot calculate SK through the $Test(W)$ query. Therefore, the probability of GM_1 is the same as the probability of GM_0:

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

GM_2: GM_2 performs more $Send(W)$ operation than GM_1. According to Zipf's law [40], the probability of GM_2 is:

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

GM_3: GM_3 performs one more $Hash(W)$ operation and one less $Send(W)$ operation than GM_2. Based on the birthday paradox, the probability of GM_3 is:

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

GM_4: In GM_4, the security of the protocol can be analyzed from two events. The first event is to acquire the TCS long-term key x, to demonstrate that the protocol can guarantee perfect forward security. The second event is to acquire the random number of any of the three entities, to demonstrate that the protocol can withstand temporary value disclosure attacks.

(1) Perfect forward security: A uses Π_{TCS}^z to acquire the TCS long-term key x or $\Pi_{U_i}^x$, $\Pi_{S_j}^y$ to obtain the private value used in the registration phase.
(2) Temporary value disclosure attacks: A uses $\Pi_{U_i}^x$, $\Pi_{S_j}^y$ or Π_{TCS}^z to obtain a random number from any of the three entities.

For the one-time event, assume that A acquired the TCS long-term key x, or the private value of U_i and S_j, during the registration phase. Because A could not calculate the value of UN_i, CN_j, and TN_{TCS}, it cannot calculate the value of SK, where $SK = h(UN_i \oplus CN_j \oplus TN_{CS})$. For the latter event, even if A can obtain UN_i, the values of CN_j and TN_{CS} are secret, so SK cannot be calculated. Similarly, even if A can obtain CN_j or TN_{CS}, the value of SK cannot be computed. Therefore, the probability of GM_4 is:

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

GM_5: In GM_5, A uses $Corrupt(W)$ to capture the parameters in SC $\{UC_i, UE_i, UF_i, UG_i, h()\}$. U_i registers with TCS using identity UID_i, password UPW_i, random number R_i, and BIO_i. A would like to guess $UE_i = (RID_i \parallel UA_i)$, but UID_i,

UPW_i, R_i and BIO_i are secret. The probability of A guessing l bits of biometric information is $1/2^l$. According to Zipf's law [40], when $q_{send} \leq 106$, the probability that A can guess the password is greater than 0.5. The proposed protocol has been shown to be resistant to offline key guessing attacks. Therefore, the probability of GM_5 is:

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

GM_6: When A employs a query $h(UN_i \oplus CN_j \oplus TN_{CS})$ in GM_6, the game is over. Check whether the proposed protocol can resist impersonation attacks. Therefore, the probability of GM_6 is:

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

Since the chances of GM_6 succeeding and failing are both 1/2, the probability that A can guess SK is:

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

We can obtain the following results using the preceding formula:

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

As a result, we can obtain

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

5.2 ProVerif

To verify the security of the proposed protocol, ProVerif is used for verification. Figure 4 (a) shows how the ProVerif code is defined. Our proposed protocol includes three entities: U_i, S_j, and TCS. The six events involved are UserAuthed, UserStarted, ControlServerAcUser, ControlServerAcCloudServer, CloudServerAcControlServer, and UserAcControlServer, as shown in Figure 4 (b). These events mark the start and finish of the protocol, as well as whether the mutual authentication of U_i, S_j, and TCS is accurate.

(1) *ch* and *sh* indicate a public channel and secure channel respectively. Registration phase messages are delivered over the secure channel, while login and authentication phase messages are delivered over the public channel. The session keys adopted are SK_i, SK_j, SK_{CS}, which respectively indicate the SK of user SK_i, cloud server SK_j, and control server SK_{CS}. We also define operations such as $h()$, $\|$, and $\oplus$. Security verification is performed using defined queries. Figure 4 (a), (b), and (c) represent the detailed description.

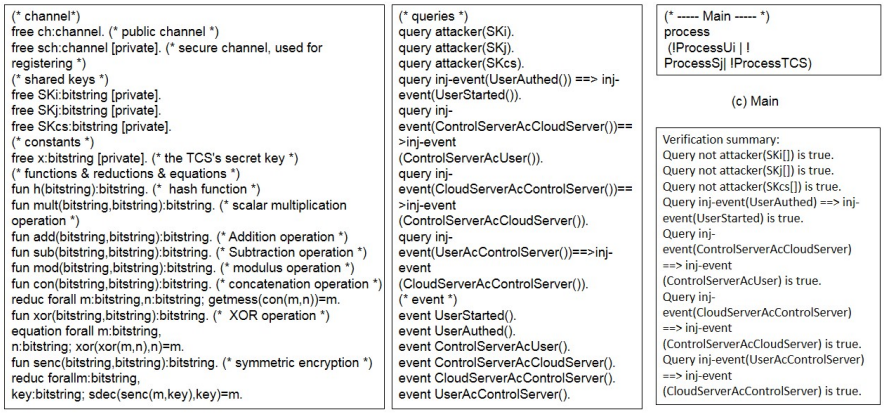

Figure 4: (a)Definitions; (b)Events; (c)Main; (d)Results.

(2) Figure 5 (a)(b) details the process for U_i and S_j.
(3) Figure 6 details the process for TCS.
(4) The verification results can be obtained from Figure 4 (d). The results prove that our protocol can successfully pass Proverif's security verification and resist attacks.

The empirical results show that our protocol can be implemented smoothly according to the above events. It can resist attacks by A, who cannot obtain parameters SK_i, SK_j, and SK_{CS}. Three entities can complete authentication and key agreement.

5.3 Informal Security Analysis

5.3.1 Privileged Insider Attacks

Suppose A can obtain CZ_j from the database of S_j. Because A is unable to retrieve the value of $CSID_j$, the value of CN_j cannot be calculated. Therefore, the value of $\{UN_i, CW_j, UN_i \oplus TN_{CS}\}$ also cannot be calculated. Thus, A cannot calculate SK, where $SK = h(UN_i \oplus TN_{CS} \oplus CN_j)$. Therefore, our protocol is resistant to privileged insider attacks.

5.3.2 Temporary Value Disclosure Attacks

Suppose A can obtain the random number for any of the entities in U_i and S_j. If A obtains the random number UN_i in U_i and uses the $\{UJ_i, UK_i\}$ transmitted over the public channel, it can calculate $h(TB_i \parallel t_i) = UJ_i \oplus UN_i$, $CSID_j = UK_i \oplus h(TB_i \parallel UN_i \parallel t_i)$. A can then calculate $\{h(TB_i \parallel t_i), CSID_j\}$, but it cannot calculate UY_i and $CN_j \oplus TN_{CS}$. Finally, A cannot calculate $SK_j = h(CN_j \oplus TN_{CS} \oplus UN_i)$. When the attacker obtains the random number CN_j in S_j, the method of obtaining

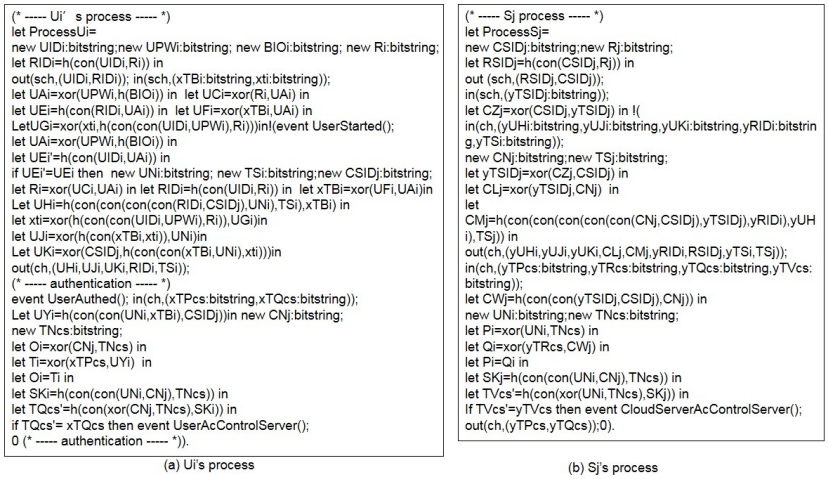

Figure 5: Process of U_i, S_j.

SK is the same as above. Therefore, our protocol is resistant to temporary value leakage attacks.

5.3.3 Anonymity and Untraceability

In our protocol, we use random numbers and hash functions to hide the real identities of U_i and S_j. During authentication, only the pseudo-identity RID_i or $CSID_j$ is used to ensure the anonymity of U_i and S_j. In addition, random numbers are different in different sessions, which also ensures that each entity is not traceable. Therefore, our protocol ensures anonymity and untraceability.

5.3.4 Replay Attacks

It is assumed that A can intercept all messages M_1, M_2, M_3, M_4 over the public channel during authentication. If A resends all the messages from the previous round, our protocol checks the freshness of the current timestamp. If the timestamp is not fresh, the session terminates. Therefore, our protocol is resistant to replay attacks.

5.4 Security and Performance Comparison

In this section, we compare our proposed protocol with Zhou et al. [41] , Kang et al. [37], and Huang et al. [38] with respect to two aspects of security and performance.

5.4.1 Security Comparison

In the security comparison, ✓ indicates that a protocol can resist an attack, and × indicates that it cannot resist the attack. Table 3 depicts the results of the security comparison analysis in detail. It can be seen from Table 3 that protocol of Zhou et

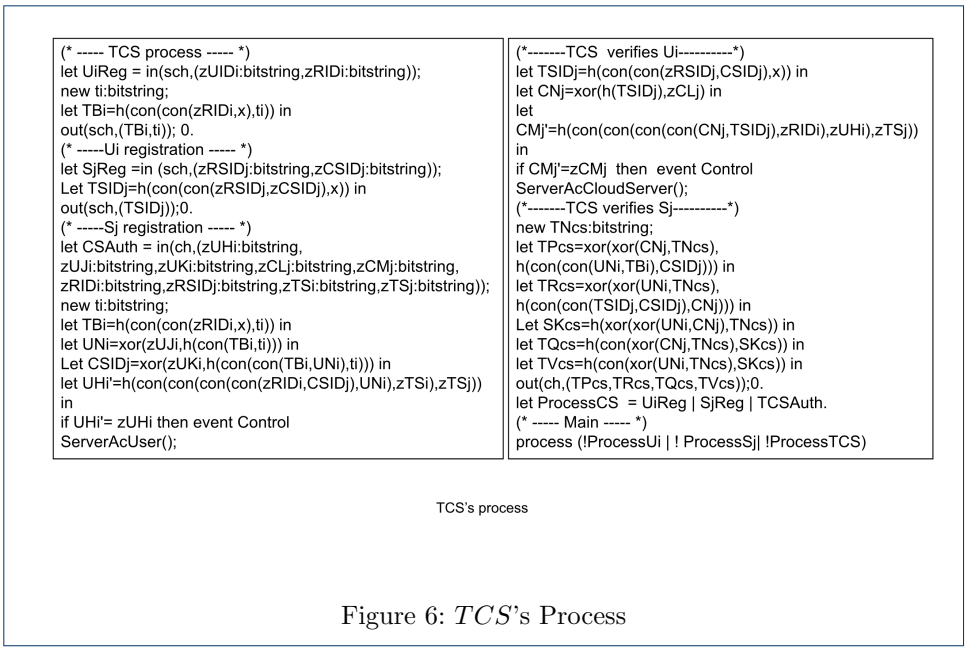

Figure 6: TCS's Process

al. [41] is unable to resist replay attacks, user impersonation attacks, and privileged insider attacks, and fails to provide mutual authentication; the protocol of Kang et al. [37] is vulnerable to off-line password guessing attacks; the protocol of Huang et al. [38] cannot resist temporary value disclosure attacks or privileged insider attacks. Our protocol is resistant to known attacks and has good security.

Table 3: Comparison of Security

Security Properties	[41]	[37]	[38]	Ours
Replay Attacks	×	✓	✓	✓
Mutual Authentication	×	✓	✓	✓
Off-line Password Guessing Attacks	✓	×	✓	✓
User Impersonation Attacks	×	✓	✓	✓
Privileged Insider Attacks	×	✓	×	✓
Temporary Value Leakage Disclosure Attacks	✓	✓	×	✓

5.4.2 Performance Comparison

Performance comparison here mainly refers to two aspects: computational cost and communication cost. We compare the performance of our protocol with other protocols. Our protocol performs only two kinds of computing operations: hash functions and XOR operations. XOR and join calculations are usually ignored because of their small cost. We used a Lenovo desktop computer and a Lenovo notebook computer running the Windows 10 operating system to implement the code. The desktop computer included an Intel(R) Core(TM) I5-9500 CPU @ 3.00ghz processor with 8 Gb of RAM to simulate the server. The notebook computer included

Table 4: Computational Cost Comparison.

protocol	U_i	S_j	TCS	Tocal
Zhou et al. [41]	$10T_h \approx 0.032ms$	$7T_h \approx 0.0224ms$	$19T_h \approx 0.0608ms$	$36T_h \approx 0.1152ms$
Kang et al. [37]	$8T_h \approx 0.0256ms$	$4T_h \approx 0.0128ms$	$11T_h \approx 0.0352ms$	$23T_h \approx 0.0736ms$
Huang et al. [38]	$8T_h \approx 0.0256ms$	$4T_h \approx 0.0128ms$	$10T_h \approx 0.032ms$	$22T_h \approx 0.0704ms$
Ours	$10T_h \approx 0.032ms$	$5T_h \approx 0.016ms$	$12T_h \approx 0.0384ms$	$27T_h \approx 0.0864ms$

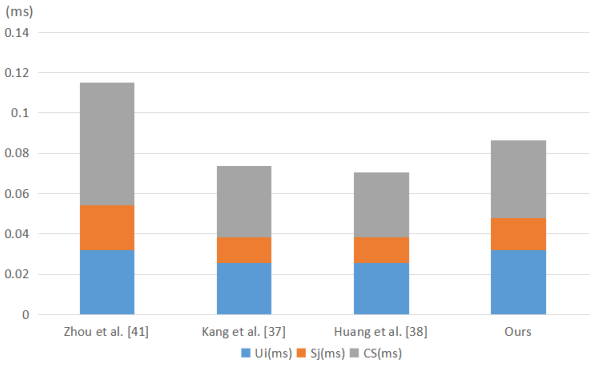

Figure 7: Computational Cost

Table 5: Communication Cost Comparison

Protocols	Rounds	Communication cost
Zhou et al. [41]	4	4704 bits
Kang et al. [37]	4	3426 bits
Huang et al. [38]	4	3232 bits
Ours	4	3232 bits

an Intel(R) Core(TM) I7-6700qh CPU @ 2.60ghz processor and with 8 G of RAM to simulate the control of the server. The implemented used IntelliJ Idea Version 2020.3. The calculation cost of the hash function was $T_h \approx 0.0032ms$. Table 4 describes the computational cost results of different protocols. Because of the increased security of our protocol, we used extra steps to realize the authentication phase for two-way authentication, so our protocol required more computation than those of Kang et al. [37] and Huang et al. [38]. The computational cost in Table 4 can be obtained from the bar chart in Figure 7.

We mainly compare the communication cost of the login and authentication phase over a public channel. We assume that it takes 32 bits to represent a timestamp, 160 bits to represent an identity or random number, and 256 bits to represent a hash function. The calculation of the communication cost of our protocol is based on these assumptions.

The communication cost of the other protocols is also calculated in the same way. Our protocol transmits four rounds of messages over a public channel, namely: $M_1 = \{UH_i, UJ_i, UK_i, RID_i, TS_i\}$, $M_2 = \{UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_i, TS_j\}$, $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$, $M_4 = \{TP_{CS}, TQ_{CS}\}$. $\{UH_i, CM_j, TQ_{CS}, TV_{CS}\}$ belongs to the hash value, $\{TS_i, TS_j\}$ belongs to the timestamp, $\{RID_i, RSID_j\}$ belongs to identities, and $\{UJ_i, UK_i, CL_j, TP_{CS}, TR_{CS}\}$ belong to random numbers. Therefore, the communication cost of our protocol is 3232 bits.

Similarly, the protocol communication cost of Zhou et al. [41] protocol, Kang et al. [37] and Huang et al. [38] are 4704 bits, 3426 bits and 3232 bits respectively. The cost of communication for each protocol is depicted in the Table 5.

According to the above comparison, it can be found that in terms of security, our protocol can withstand known attacks, while the other protocols cannot. As a result, our protocol is more secure than other protocols. In terms of calculation cost, although the proposed protocol is more expensive than the protocol of Kang et al. [37] and Huang et al. [38], it is much lower than the protocol of Zhou et al. [41]. In terms of communication cost, the proposed protocol is lower than that proposed by Zhou et al. [41] and Kang et al. [37], and the cost is the same as the protocol of Huang et al. [38].We use the ROR model for formal analysis, which proves that our protocol is secure.

The above results prove that our protocol is more suitable for the limited memory and data-intensive environment of IoT devices in distributed cloud computing architecture. In addition, we hope to propose secure and efficient authentication protocols for other intelligent computing applications in the future.

6 Conclusions

We described the need for the combination of IoT and cloud computing, then reviewed the AKA protocol supporting the cloud computing environment of the IoT. Secondly, we briefly describe the protocol of Huang et al. [38], and point out that Huang et al. [38]'s protocol is unable to resist privileged insider attacks or temporary value disclosure attacks. Finally, we proposed an enhanced security protocol to solve the security defects of Huang et al. [38]. By using the ROR model for formal analysis, we proved that our protocol is secure. Our protocol security was verified using the ProVerif tool and informal analysis. Finally, our protocol was compared with other protocols in terms of performance, and the findings demonstrate that our protocol outperforms them.

Abbreviations

IoT: Internet of Things; ROR: real-oracle-random; AKA: authentication and key agreement SaaS: software as a service; PaaS: platform as a service; IaaS: infrastructure as a service; REID: radio frequency identification devices; TCS: control server; SC: smart card; A: the attacker; SK: session key;

Acknowledgements

Not applicable.

Funding

The authors received no specific funding for this study.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

Conceptualization, Chien-Ming Chen and Saru Kumari; methodology, Tsu-Yang Wu and Fangfang Kong; software, Qian Meng; formal analysis, Fangfang Kong and Saru Kumari ; investigation, Chien-Ming Chen and Qian Meng; writing—original draft preparation, Tsu-Yang Wu and Fangfang Kong. All authors have read and agreed to the published version of the manuscript.

Author details

[1]College of Computer Science and Engineering, Shandong University of Science and Technology, 266590, Qingdao, China. [2]Department of Mathematics, Chaudhary Charan Singh University, Uttar Pradesh 250004, Meerut, India.

References

1. Xue, X., Wu, X., Jiang, C., Mao, G., Zhu, H.: Integrating sensor ontologies with global and local alignment extractions. Wireless Communications and Mobile Computing **2021**, 6625184 (2021)
2. Meng, Z., Pan, J.-S., Tseng, K.-K.: Pade: An enhanced differential evolution algorithm with novel control parameter adaptation schemes for numerical optimization. Knowledge-Based Systems **168**, 80–99 (2019)
3. Xue, X., Zhang, J.: Matching large-scale biomedical ontologies with central concept based partitioning algorithm and adaptive compact evolutionary algorithm. Applied Soft Computing **106**, 107343 (2021)
4. Pan, J.-S., Liu, N., Chu, S.-C., Lai, T.: An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems. Information Sciences **561**, 304–325 (2021)
5. Chaudhry, S.: Combating identity de-synchronization: an improved lightweight symmetric key based authentication scheme for iov. Journal of Network Intelligence **6**, 656–667 (2021)
6. Xiong, H., Chen, J., Mei, Q., Zhao, Y.: Conditional privacy-preserving authentication protocol with dynamic membership updating for vanets. IEEE Transactions on Dependable and Secure Computing (01), 1–1 (2020)
7. Wu, T.-Y., Lee, Y.-Q., Chen, C.-M., Tian, Y., Al-Nabhan, N.A.: An enhanced pairing-based authentication scheme for smart grid communications. Journal of Ambient Intelligence and Humanized Computing, 1–13 (2021)
8. Luo, Y., Zheng, W., Chen, Y.-C.: An anonymous authentication and key exchange protocol in smart grid. Journal of Network Intelligence **6**(2), 206–215 (2021)
9. 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. Security and Communication Networks **2021**, 6658041 (2021)
10. 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 Transactions on Cloud Computing **2021**, 1–1 (2021)
11. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 598–609 (2007)
12. Chandra, S., Yafeng, W.: Cloud things construction—the integration of internet of things and cloud computing. Future Generation Computer Systems **56**(C), 684–700 (2016)
13. Mushtaq, M.F., Akram, U., Khan, I., Khan, S.N., Shahzad, A., Ullah, A.: Cloud computing environment and security challenges: A review. International Journal of Advanced Computer Science and Applications **8**(10), 183–195 (2017)
14. Pan, J.-S., Sun, X.-X., Chu, S.-C., Abraham, A., Yan, B.: Digital watermarking with improved sms applied for qr code. Engineering Applications of Artificial Intelligence **97**, 104049 (2021)
15. 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 Networks **20**, 96–112 (2014)
16. Farash, M.S., Turkanović, M., Kumari, S., Hölbl, M.: An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment. Ad Hoc Networks **36**, 152–176 (2016)
17. Amin, R., Biswas, G.: A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks. Ad Hoc Networks **36**, 58–80 (2016)
18. Braeken, A.: Puf based authentication protocol for iot. Symmetry **10**(8), 352 (2018)
19. Panda, P.K., Chattopadhyay, S.: A secure mutual authentication protocol for iot environment. Journal of Reliable Intelligent Environments **6**(2), 79–94 (2020)
20. Challa, S., Wazid, M., Das, A.K., Kumar, N., Reddy, A.G., Yoon, E.-J., Yoo, K.-Y.: Secure signature-based authenticated key establishment scheme for future iot applications. Ieee Access **5**, 3028–3043 (2017)
21. Liu, H., Ning, H., Xiong, Q., Yang, L.T.: Shared authority based privacy-preserving authentication protocol in cloud computing. IEEE Transactions on parallel and distributed systems **26**(1), 241–251 (2014)
22. Kalra, S., Sood, S.K.: Secure authentication scheme for iot and cloud servers. Pervasive and Mobile Computing **24**, 210–223 (2015)
23. 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 Generation Computer Systems **78**, 1005–1019 (2018)
24. Xue, K., Hong, P., Ma, C.: A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture. Journal of Computer and System Sciences **80**(1), 195–206 (2014)
25. Chuang, M.-C., Chen, M.C.: An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics. Expert Systems with Applications **41**(4), 1411–1418 (2014)
26. 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. Wireless Communications and Mobile Computing **2018**, 3048697 (2018)
27. Irshad, A., Ahmad, H.F., Alzahrani, B.A., Sher, M., Chaudhry, S.A.: An efficient and anonymous chaotic map based authenticated key agreement for multi-server architecture. KSII Transactions on Internet and Information Systems (TIIS) **10**(12), 5572–5595 (2016)
28. Wu, F., Li, X., Xu, L., Sangaiah, A.K., Rodrigues, J.J.: Authentication protocol for distributed cloud computing: An explanation of the security situations for internet-of-things-enabled devices. IEEE Consumer Electronics Magazine **7**(6), 38–44 (2018)
29. Fan, K., Luo, Q., Zhang, K., Yang, Y.: Cloud-based lightweight secure rfid mutual authentication protocol in iot. Information Sciences **527**, 329–340 (2020)
30. Yu, Y., Hu, L., Chu, J.: A secure authentication and key agreement scheme for iot-based cloud computing environment. Symmetry **12**(1), 150 (2020)
31. He, D., Zeadally, S., Kumar, N., Wu, W.: Efficient and anonymous mobile user authentication protocol using

self-certified public key cryptography for multi-server architectures. IEEE transactions on information forensics and security **11**(9), 2052–2064 (2016)
32. Kumar, V., Ahmad, M., Mishra, D., Kumari, S., Khan, M.K.: Rseap: Rfid based secure and efficient authentication protocol for vehicular cloud computing. Vehicular Communications **22**, 100213 (2020)
33. Wazid, M., Das, A.K., Bhat, V., Vasilakos, A.V.: Lam-ciot: Lightweight authentication mechanism in cloud-based iot environment. Journal of Network and Computer Applications **150**, 102496 (2020)
34. Irshad, A., Chaudhry, S.A., Alomari, O.A., Yahya, K., Kumar, N.: A novel pairing-free lightweight authentication protocol for mobile cloud computing framework. IEEE Systems Journal **15**(3), 3664–3672 (2020)
35. Rangwani, D., Om, H.: A secure user authentication protocol based on ecc for cloud computing environment. Arabian Journal for Science and Engineering **46**(4), 3865–3888 (2021)
36. Zargar, S., Shahidinejad, A., Ghobaei-Arani, M.: A lightweight authentication protocol for iot-based cloud environment. International Journal of Communication Systems **34**(11), 4849 (2021)
37. Kang, B., Han, Y., Qian, K., Du, J.: Analysis and improvement on an authentication protocol for iot-enabled devices in distributed cloud computing environment. Mathematical Problems in Engineering **2020**, 3048697 (2020)
38. 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 Journal on Wireless Communications and Networking **2021**(1), 1–21 (2021)
39. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. Journal of the ACM (JACM) **51**(4), 557–594 (2004)
40. Wang, D., Cheng, H., Wang, P., Huang, X., Jian, G.: Zipf's law in passwords. IEEE Transactions on Information Forensics and Security **12**(11), 2776–2791 (2017)
41. Zhou, L., Li, X., Yeh, K.-H., Su, C., Chiu, W.: Lightweight iot-based authentication scheme in cloud computing circumstance. Future generation computer systems **91**, 244–251 (2019)

Figure Title and Legend

Figure 1: S_j Registration Phase. The process of cloud server registration.
Figure 2: U_i Registration Phase. The process of user registration.
Figure 3: Login and Authentication Phase. The process of user and cloud server authentication and session key establishment.
Figure 4: (a) Definitions. It mainly defines communication channels, constant, various variables, and functions.
Figure 4: (b) Events. It defines various events and queries.
Figure 4: (c) Main. It define the main function.
Figure 4: (d) Results. It shows the execution results of ProVerif.
Figure 5: (a) U_i's process. To describe the user execution process in ProVerif.
Figure 5: (b) S_j's process. To describe the cloud server execution process in ProVerif.
Figure 6: TCS's Process. To describe the control server execution process in ProVerif.
Figure 7: Computational Cost. The compared results between the recent state-of-art protocols with our protocol.