Wu *et al. J Wireless Com Network* (2023) 2023:36
https://doi.org/10.1186/s13638-023-02245-4

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

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, Qingdao 266590, China
[2] Department of Mathematics, Chaudhary Charan Singh University, Meerut, Uttar Pradesh 250004, India

Abstract

With the continuous progress of the Internet of Things (IoT) technology, IoT devices have gradually penetrated all aspects of life. However, due to rapid data growth, IoT devices with limited memory resources cannot store massive data. Cloud computing is an Internet-centric network that can provide security services and data storage. The combination of IoT devices with cloud computing considerably promotes resource sharing, facilitates users' access to information at any time, and provides users with corresponding cloud computing services on time. Because the information transmitted through public channels is easily intercepted, tampered with, and eavesdropped on by malicious attackers. As a result, users' and servers' private information are disclosed. Numerous scholars have proposed different authentication protocols in this environment to protect the communications between users and servers. Amin et al. devised a smart card based authentication protocol. Unfortunately, Kang et al. demonstrated that their protocol was insecure. Huang et al. discovered Kang et al.'s improvement also has security flaws and then designed a protocol to enhance security. In this paper, we first show that Huang et al.'s protocol cannot resist privileged insider and temporary value disclosure attacks. Afterward, we propose an enhanced protocol based on their protocol. Finally, we use formal/informal security analysis to demonstrate the security of the improved protocol. The comparison results are indicated that our protocol has lower computational and communication costs under the same security level.

Keywords: IoT, Cloud computing, Authentication, Cryptanalysis

1 Introduction

The Internet of Things (IoT) [1–3] is a network that combines different sensor devices with the Internet. Following the development of the Internet and artificial intelligence technology [4–6], IoT technology is progressively being used in many different fields such as vehicle [7–9], smart grid [10, 11], healthcare [12–14], smart home [15, 16], smart manufacturing, and etc. As a subset of the IoT technology, the Industrial Internet of Things (IIoT) [17] has become a synonym for smart manufacturing. With the explosion of data, memory-constrained IoT devices are challenged, and cloud/fog computing [18–20] comes into being.

The IoT-enabled devices in the distributed cloud computing architecture significantly promote resource sharing, facilitate users to access information anytime, and provide users with the corresponding IoT services. In a distributed cloud computing environment, messages are transmitted through open and unstable wireless channels, making them easy for malicious attackers to intercept, tamper, and eavesdrop. Authentication and key agreement (AKA) is a cryptographic primitive. AKA can be used to realize secure communication through mutual authentication between entities and negotiation of a session key. In recent years, numerous scholars have put forth various authentication protocols for IoT-enabled devices in the cloud computing environment. However, designing a secure authentication protocol in this environment is still a challenge.

In 2014, Turkanovic et al. [21] devised an IoT-based authentication protocol. Their protocol claimed lightweight and provably secure to achieve secure communications between user and sensor node. Nevertheless, Farash et al. [22] indicated that Turkanovic et al.'s protocol could not withstand man-in-the-middle (MITM) attacks and could not guarantee the anonymity of sensor node. Then, they designed a new AKA protocol. Unfortunately, Amin et al. [23] proved that Farash et al.'s protocol also had several security flaws including user impersonation and stolen smart card (SSC) attacks as well as violating anonymity. To overcome the shortcomings of their protocol, Amin et al. devised a secure AKA protocol that guaranteed the anonymity of entities. Chatterjee et al. [24] put forward a secure AKA protocol employing physical unclonable functions (PUF) in the IoT environment and purported that the protocol can withstand known attacks. Unfortunately, Braeken [25] found that Chatterjee et al.'s protocol could not resist MITM, denial of service (DoS), and replay attacks. Panda and Chattopadhyay [26] presented a secure AKA protocol using elliptic curve cryptosystems (ECC). Bao et al. [27] proposed an authentication protocol using an attribute-based signature (ABS) concept for the IIoT. The ABS used in this protocol can provide privacy-preserving authentication for IIoT. Chithaluru et al. [28] proposed a radio scheme for the IoT, which can adapt to the environment well and provide reliable communication.

In 2014, Liu et al. [29] devised an AKA protocol that used shared permissions in the cloud. Their protocol provided privacy-preserving for user and cloud server based on attributes and sharing permissions. Kalra et al. [30] designed an AKA protocol using ECC in the IoT and cloud environment. Amin et al. [31] pointed out security vulnerabilities in Xue et al.'s protocol [32] and Chuang et al.'s protocol [33]. They also devised an AKA protocol in the cloud. However, Wang et al. [34] indicated that Amin et al.'s protocol cannot withstand offline dictionary attacks and could not provide anonymity. They then designed a secure ECC-based AKA protocol. Wu et al. [35] presented a novel chaotic-based AKA protocol. Unfortunately, Wang et al. [36] indicated that Wu et al.'s protocol could not resist temporary value disclosure (TVD) and SSC attacks. Fan et al. [37] put forward an AKA protocol using radio frequency identification (RFID) technology for the cloud computing environment. He et al. [38] put forward an anonymous authentication protocol employing asymmetric cryptography without the registration authority. However, Yu et al. [39] found that He et al.'s protocol suffered from insider and DoS attacks, and designed an improvement. Irshad et al. [40] designed a new type of lightweight AKA protocol without pairings. Rangwani et al. [41] presented an enhanced AKA protocol to overcome the shortcomings in [42]. Zhou et al. [43] devised a two-factor provably secure AKA protocol. Unfortunately, Wang et al. [44] indicated that Zhou et al.'s protocol could not withstand impersonation and TVD attacks, as well as violated perfect forward secrecy (PFS). Martinez-Pelaez et al. [45] also stated that Zhou et al.'s protocol is insecure against MITM attacks and violated mutual authentication (MA). However, Yu et al. [46] demonstrated that the improved protocol [45] could not resist replay and session key disclosure (SKD) attacks as well as violated MA.

In 2020, Kang et al. [47] conducted a cryptanalysis of [31], demonstrating that their protocol was insure. Subsequently, they devised an improved AKA protocol. Wu et al. [48] also proposed an improved AKA protocol to enhance the security of [43]. Recently, Huang et al. [49] stated that [47] was unable to withstand offline password guessing (OPG) attacks and improved the protocol using lightweight operations. Some related authentication protocols in distributed cloud computing environments are summarized in Table 1. In this paper, we first analyze [49] and find some security weaknesses. Then, we based on their protocol to propose an improvement. Our contributions are summarized as follows.

(1) We first find Huang et al.'s protocol is insecure against privileged insider and TVD attacks. In order to enhance the shortcomings of [49], we propose an improved protocol.
(2) We use the real or random (ROR) model (formal security analysis), informal security analysis, and ProVerif to verify the security of our improved protocol. The results are indicated that our protocol is provably secure to ensure secure communication between entities.

Table 1 Overview of authentication protocols

Protocols	Adopted cryptographic operations	Security weaknesses
Chuang et al. [33]	Hash function	User impersonation attacks SKD attacks
Amin et al. [31]	Hash function Smart card	Offline dictionary attacks User anonymity
Wu et al. [35]	Hash function Chaotic maps Smart card	SSC attacks TVD attacks
Zhou et al. [43]	Hash function	TVD attacks Impersonation attacks SKD attacks Replay attacks PFS
Martinez-Pelaez et al. [45]	Hash function Symmetric encryption/decryption	SKD attacks Replay attacks MA
Fan et al. [37]	RFID Symmetric encryption/decryption	–
Kang et al. [47]	Hash function	OPG attacks
Wu et al. [48]	Hash function Smart card	–
Rangwani et al. [41]	ECC Hash function	–

(3) Finally, comparing the proposed protocol with other protocols in terms of security and performance, the results are indicated that our protocol has lower computational and communication costs as well as resisting well-known attacks.

The remainder of this paper is structured as follows: we summarize the adopted methods and experiments in Sect. 2. We briefly review Huang et al.'s protocol and proved that their protocol has two security problems in Sect. 3.

In Sect. 4, we describe the details of our proposed protocol.

Results and discussion are made in Sect. 5.

We draw the conclusion in Sect. 6.

2 Methods and experiments

We adopt the same architecture of [49] in our protocol. There are three entities: User U_i, cloud server S_j, and control server TCS, where U_i can obtain services from S_j by using IoT devices, S_j provides several services on demand of U_i, and TCS is in charge of U_i and S_j registration and authentication. Before the authentication phase, U_i and S_j register with TCS. After finishing this phase, legitimate U_i and S_j can authenticate each other and negotiate a common key with the help of TCS. Our protocol only performs two kinds of lightweight operations: hash and XOR operations.

The security of our proposed protocol is verified by (1) using ROR model for formal analysis; (2) using a verification software, ProVerif, on Lenovo desktop with Windows 10 operating system; (3) using informal security analysis for some specific security requirements. Finally, we compare other related authentication protocols with our protocol in terms of security and performance. The results are shown that our protocol is feasible and has a lower cost.

3 Review and cryptanalysis of Huang et al.'s protocol

In this section, we review Huang et al.'s protocol [49] and prove that their protocol is insecure. Their protocol includes three entities: user U_i, cloud server S_j, and control server TCS. Three phases of "registration," "login," and "authentication and key agreement" are briefly reviewed below. Some important symbols used in this paper are shown in Table 2.

Table 2 Notations

Notations	Description
$CSID_j$	S_j's identity
UID_i	U_i's identity
UPW_i	U_i's password
BIO_i	U_i's biometric
x	TCS's private secret key

3.1 Review Huang et al.'s protocol

3.1.1 Registration phase

Registration of S_j.

(1) S_j chooses $CSID_j$ and a random number R_j, then though the secure channel sends $\{CSID_j, R_j\}$ to TCS.
(2) After receiving $\{CSID_j, R_j\}$, 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.
(3) After receiving $\{RSID_j\}$, S_j stores secret parameter $\{RSID_j, R_j\}$ in memory.

Registration of U_i.

(1) U_i selects UID_i, UPW_i, BIO_i, and R_i to compute $RID_i = h(UID_i \parallel R_i)$, $UA_i = UPW_i \oplus h(BIO_i)$, and $UC_i = R_i \oplus UA_i$. Next, U_i sends $\{UID_i, RID_i, UA_i\}$ to TCS.
(2) After receiving $\{UID_i, RID_i, UA_i\}$ sent by U_i, TCS first verifies 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). Finally, TCS sends SC to U_i.
(3) After receiving SC, U_i stores UC_i in it. 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.

(1) U_i generates UN_i and 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 $|TS_j - TS_i| \leq \Delta T$. If the timestamp is valid, S_j generates CN_j and TS_j. Then, S_j 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 $|TS_{SC} - TS_j| \leq \Delta T$. If the timestamp is invalid, 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 $TP_{CS} = CN_j \oplus TN_{CS} \oplus h(UN_i \parallel TE_i \parallel UH_i)$, $TR_{CS} = UN_i \oplus TN_{CS} \oplus h(RSID_j \parallel CN_j)$, $SK_{CS} = h(UN_i \oplus CN_j \oplus TN_{CS})$, $TQ_{CS} = h((UN_i \oplus TN_{CS}) \parallel SK_{CS})$, and $TV_{CS} = h((UN_i \oplus TN_{CS}) \parallel SK_{CS})$. 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 establish the session key SK, where $SK = h(UN_i \oplus CN_j \oplus TN_{CS})$.

3.4 Cryptanalysis of Huang et al.'s protocol

Here, we show that their protocol [49] is insecure against (1) privileged insider and (2) TVD attacks.

3.4.1 Adversary model

D-Y [50] and C-K [51] adversarial models are well used. We follow both models to describe the attacker (A)'s capabilities

(1) A is capable of intercepting, deleting, or eavesdropping messages transmitted in the public channel.

(2) A is able to intercept the temporary value generated by user or cloud server in each session.

(3) Through power analysis, A is capable of acquiring the data stored in the smart card.

(4) A is capable of obtaining the parameters stored in the cloud server.

3.4.2 Privileged insider attacks

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

(1) A can capture $M_2 = \{CK_j, CL_j, TSID_j, UG_i, UH_i, UJ_i, RID_i, TS_i, TS_j\}$ transmitted in the public channel, then get the values of CN_j and CW_j, where $CN_j = CK_j \oplus RSID_j$ and $CW_j = h(RSID_j \parallel CN_j)$.

(2) A can capture the message $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$ transmitted in the public channel, and can calculate the value $UN_i \oplus TN_{CS}$ though $UN_i \oplus TN_{CS} = TR_{CS} \oplus CW_j$.

(3) At last, A can successfully calculate SK, where $SK = h(UN_i \oplus TN_{CS} \oplus CN_j)$.

Therefore, Huang et al.'s protocol is insecure against privileged insider attacks.

3.4.3 Temporary value conflict of interest (TVD) attacks

Assume that A can obtain the temporary value of each entity. Then, we describe the process of A successfully calculating SK in the two cases.

Case I. A can obtain UN_i of U_i.

(1) A can capture $M_2 = \{CK_j, CL_j, TSID_j, UG_i, UH_i, UJ_i, RID_i, TS_i, TS_j\}$ transmitted in the public channel, then get the values of TE_i and UZ_i, where $TE_i = UH_i \oplus UN_i$ and $UZ_i = h(UN_i \parallel TE_i \parallel UH_i)$.
(2) A can capture $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$ transmitted in the public channel, and can calculate the value $CN_j \oplus TN_{CS}$ though $CN_j \oplus TN_{CS} = TP_{CS} \oplus UZ_i$.
(3) Finally, A can successfully calculate SK, where $SK = h(UN_i \oplus TN_{CS} \oplus CN_j)$.

Case II. A obtains CN_j of S_j.

(1) A can capture $M_2 = \{CK_j, CL_j, TSID_j, UG_i, UH_i, UJ_i, RID_i, TS_i, TS_j\}$ transmitted in the public channel, then get the values of $RSID_j$ and CW_j, where $RSID_j = CK_j \oplus CN_j$ and $CW_j = h(RSID_j \parallel CN_j)$.
(2) A can capture $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$ transmitted in the public channel, and can calculate the value $UN_i \oplus TN_{CS}$ though $UN_i \oplus TN_{CS} = TR_{CS} \oplus CW_j$.
(3) Finally, A can successfully calculate SK, where $SK = h(UN_i \oplus TN_{CS} \oplus CN_j)$.

In both cases, their protocol [49] cannot resist TVD attacks.

4 Proposed protocol

In this section, we followed Huang et al.'s framework [49] to propose an improvement. Note that our enhancements are marked in red for each phase depicted in Figures 1, 2, and 3.

4.1 New registration of S_j

The new registration of S_j is shown in Fig. 1.

(1) S_j chooses $CSID_j$ and R_j to calculate $RSID_j = h(CSID_j \parallel R_j)$. Finally, S_j sends $\{RSID_j, CSID_j\}$ to TCS.

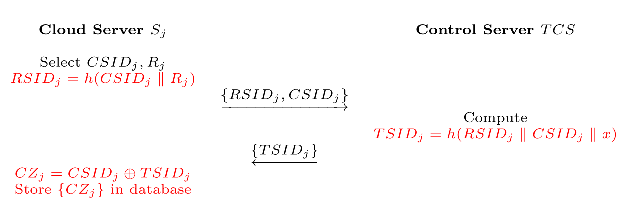

Fig. 1 New registration of S_j

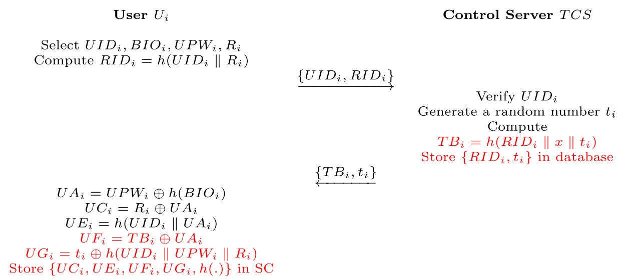

Fig. 2 New registration of U_i

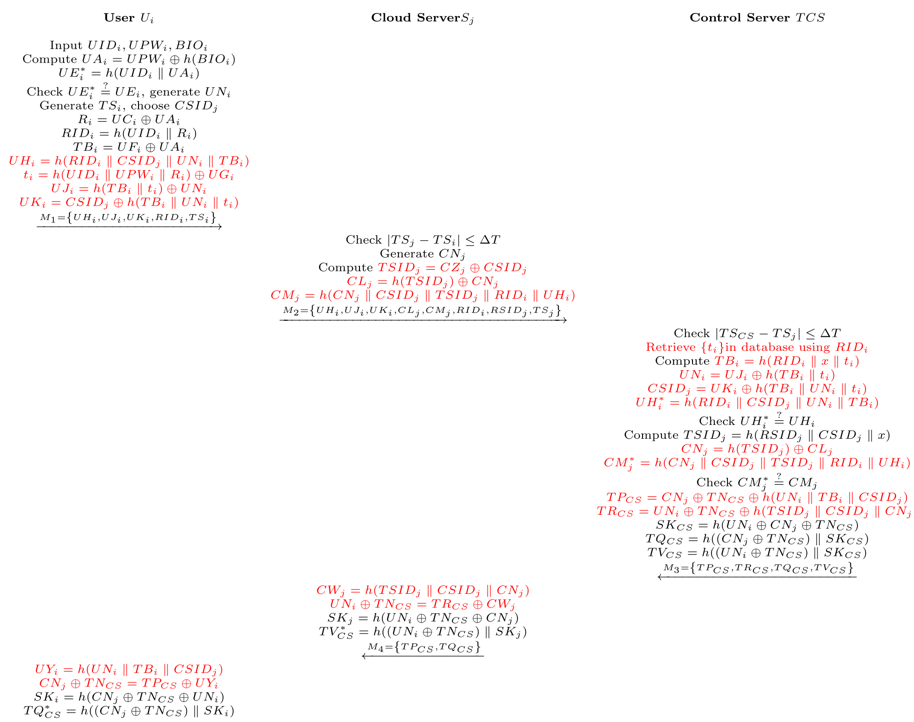

Fig. 3 New login and authentication phase

(2) After TCS receives $\{RSID_j, CSID_j\}$, it computes $TSID_j = h(RSID_j \parallel CSID_j \parallel x)$. Fianlly, TCS sends $TSID_j$ 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 memory.

4.2 New registration of U_i

The new registration of U_i is shown in Fig. 2.

(1) U_i chooses UID_i, UPW_i, BIO_i, and R_i to compute $RID_i = h(UID_i \parallel R_i)$. Then, it 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 this registration request. Otherwise, TCS chooses t_i and calculates $TB_i = h(RID_i \parallel x \parallel t_i)$. Then, TCS stores $\{RID_i, t_i\}$ in its database. Finally, TCS sends $\{TB_i, t_i\}$ to U_i.
(3) U_i receives $\{TB_i, t_i\}$ from TCS to compute $UA_i = UPW_i \oplus h(BIO_i), UC_i = R_i \oplus UA_i$, $UE_i = h(RID_i \parallel UA_i)$, $UF_i = TB_i \oplus UA_i$, 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.

4.3 New login and authentication phase

The new login and authentication phase is shown in Fig. 3.

(1) U_i inputs UID_i, UPW_i, and BIO_i to compute $UA_i = UPW_i \oplus h(BIO_i)$, $UE_i^* = h(UID_i \parallel UA_i)$. Then, U_i checks whether $UE_i^* \stackrel{?}{=} UE_i$. If equal, U_i chooses UN_i, TS_i, and S_j's identity $CSID_j$ to compute $R_i = UC_i \oplus UA_i$, $RID_i = h(UID_i \parallel R_i)$, $TB_i = UF_i \oplus UA_i$, $UH_i = h(RID_i \parallel CSID_j \parallel UN_i \parallel TB_i)$, $t_i = h(UID_i \parallel UPW_i \parallel R_i) \oplus UG_i$, $UJ_i = h(TB_i \parallel t_i) \oplus UN_i$, and $UK_i = CSID_j \oplus h(TB_i \parallel UN_i \parallel t_i)$. 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 $|TS_j - TS_i| \le \Delta T$. If valid, S_j chooses CN_j, and computes $TSID_j = CZ_j \oplus CSID_j$, $CL_j = h(TSID_j) \oplus CN_j$, $CM_j = h(CN_j \parallel CSID_j \parallel TSID_j \parallel RID_i \parallel UH_i)$. Finally, S_j sends $M_2 = \{UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j\}$ to TCS.
(3) After receiving M_2, TCS checks whether $|TS_{CS} - TS_j| \le \Delta T$. If the timestamp is valid, TCS retrieves $\{t_i\}$ from its database using RID_i. TCS computes $TB_i = h(RID_i \parallel x \parallel t_i), UN_i = UJ_i \oplus h(TB_i \parallel t_i), CSID_j = UK_i \oplus h(TB_i \parallel UN_i \parallel t_i)$, and $UH_i^* = h(RID_i \parallel CSID_j \parallel UN_i \parallel TB_i)$. 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 $TSID_j = h(RSID_j \parallel CSID_j \parallel x)$, $CN_j = h(TSID_j) \oplus CL_j$, and $CM_j^* = h(CN_j \parallel CSID_j \parallel TSID_j \parallel RID_i \parallel UH_i)$. TCS verifies the identity of S_j by comparing $CM_j^* \stackrel{?}{=} CM_j$. If equal, the identity of S_j is valid. Then, TCS chooses TN_{CS} and computes $TP_{CS} = CN_j \oplus TN_{CS} \oplus h(UN_i \parallel TB_i \parallel CSID_j)$, $TR_{CS} = UN_i \oplus TN_{CS} \oplus h(TSID_j \parallel CSID_j \parallel CN_j)$, $SK_{CS} = h(UN_i \oplus CN_j \oplus TN_{CS})$, $TQ_{CS} = h((CN_j \oplus TN_{CS}) \parallel SK_{CS})$, and $TV_{CS} = h((UN_i \oplus TN_{CS}) \parallel SK_{CS})$. Finally, TCS sends $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$ to S_j.
(4) After receiving M_3, S_j computes $CW_j = h(TSID_j \parallel CSID_j \parallel CN_j)$, $UN_i \oplus TN_{CS} = TR_{CS} \oplus CW_j$, $SK_j = h(UN_i \oplus TN_{CS} \oplus CN_j)$, and $TV_{CS}^* = h((UN_i \oplus TN_{CS}) \parallel SK_j)$. 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 $UY_i = h(UN_i \parallel TB_i \parallel CSID_j)$, $CN_j \oplus TN_{CS} = TP_{CS} \oplus UY_i$, $SK_i = h(CN_j \oplus TN_{CS} \oplus UN_i)$, and $TQ_{CS}^* = h((CN_j \oplus TN_{CS}) \parallel SK_i)$. If they are not equal, the authentication fails.

Otherwise, U_i successfully authenticates TCS and S_j. Finally, U_i, S_j and TCS achieve authentication and establish SK.

5 Results and discussion

5.1 Formal security analysis

Canetti et al. [52, 53] proposed the ROR model, which mainly judges the security of authentication protocol according to the probability that successfully breaks SK through a sequence of games. Here, we use this approach to calculate the probability of A cracking SK and prove that our protocol is secure.

5.1.1 Security model

Here, $\Pi^x_{U_i}$, $\Pi^y_{S_j}$, and Π^z_{TCS} denote the x-th instance of user U_i, y-th instance of cloud server S_j, and z-th instance of control server TCS, respectively. A can perform the following queries to $W = \{\Pi^x_{U_i}, \Pi^y_{S_j}, \Pi^z_{TCS}\}$.

(1) *Execute*(W): If A performs this query, it can obtain all transmitted messages in the previous sessions between $\Pi^x_{U_i}$, $\Pi^y_{S_j}$, and Π^z_{TCS}.
(2) *Send*(W, M): If A performs this query with M to W, the oracle in W returns a response.
(3) *Hash*(M): If A performs this query, it can acquire the corresponding hash value by entering a string M.
(4) *Corrupt*(W): If A performs this query, it will retrieve the private data of one entity, such as long-term keys, data stored in SC, or temporary values.
(5) *Test*(W): If A performs this query, a coin C is flipped. If $C = 1$, A successfully guess the correct SK. If $C = 0$, indicating that the coin is down, then A obtains a string with the same length of SK.

5.1.2 Security proof

Theorem 1 *In the ROR model, suppose that an attacker A performs Execute, Send, Hash, Corrupt, and Test queries. Then, the advantage (probability) of A that successfully guess the correct session key (break the proposed protocol P) within the polynomial time* ξ *is stated by* $Adv^{\mathcal{P}}_{A}(\xi) \leq q_{\text{send}}/2^{l-2} + q^2_{\text{hash}}/2^{l-1} + 2max\{C' \cdot q^{s'}_{\text{send}}, q_{\text{send}}/2^l\}$, *where* q_{send} *and* q_{hash} *indicate the times of the Send and Hash queries, respectively,* l *indicates the bit length of biometric, and* C' *and* s' *refer to two constants.*

1 *Proof*

Let GM_i *be the game and* $\text{Succ}^{\text{GM}_i}_{A}(\xi)$ *be the probability of A wins in* GM_i*. Here, we adopt a sequential six games to simulate the real execution of our protocol. The specific queries are shown in Table 3.*

Table 3 Simulation of oracles

On a query Send($\Pi_{U_i}^x$, *start*), $\Pi_{U_i}^x$ chooses $UN_i, TS_i, CSID_j$ to compute $R_i, RID_i, TB_i, UH_i, t_i, UJ_i, UK_i$. Then, the query is returned $M_1 = \{UH_i, UJ_i, UK_i, RID_i, TS_i\}$.
On a query Send($\Pi_{S_j}^y$, $(UH_i, UJ_i, UK_i, RID_i, TS_i)$), $\Pi_{S_j}^y$ selects CN_j, TS_j to compute $TSID_j, CL_j, CM_j$. Then, the query is returned $M_2 = \{UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j\}$.
On a query Send(Π_{CS}^z, $((UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j)$), Π_{CS}^z computes $TB_i, UN_i, CSID_j, UH_i^*$ to check UH_i^*. If true, it continues to calculate $TSID_j, CN_j, CM_j^*$, and checks CM_j^*. If true, Π_{CS}^z chooses TN_{CS} to compute $TP_{CS}, TR_{CS}, SK_{CS}, TQ_{CS}, TV_{CS}$. Then, the query is returned $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$.
On a query Send($\Pi_{S_j}^y$, $(TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS})$), $\Pi_{S_j}^y$ computes $CW_j, UN_i \oplus TN_{CS}, SK_j, TV_{CS}^*$ to check TV_{CS}^*. If the verification does not equal, it will be terminated. Otherwise, the query is returned $M_4 = \{TP_{CS}, TQ_{CS}\}$.
On a query Send($\Pi_{U_i}^x$, (TP_{CS}, TQ_{CS})), $\Pi_{U_i}^x$ verifies TQ_{CS}. If the verifications holds, $\Pi_{U_i}^x$ returns true. Otherwise, it terminates.
On a query *Execute*, we use Send queries to simulate it.
$(UH_i, UJ_i, UK_i, RID_i, TS_i) \longleftarrow$ Send($\Pi_{U_i}^x$, *start*),
$(UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j) \longleftarrow$ Send($\Pi_{S_j}^y$, $(UH_i, UJ_i, UK_i, RID_i, TS_i)$),
$(TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}) \longleftarrow$ Send(Π_{CS}^z, $(UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j)$),
$(TP_{CS}, TQ_{CS}) \longleftarrow$ Send($\Pi_{S_j}^y$, $TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}$)). This query is returned $(UH_i, UJ_i, UK_i, RID_i, TS_i)$, $(UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j)$, $(TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS})$, and (TP_{CS}, TQ_{CS}).
For a *Hash*(*M*) query, it returns a random value *h*. Note that a record (*M*, *h*) is required in the query.

GM_0: We start GM_0 by flipping C. In this game, no query is performed. Therefore, we have

$$\mathrm{Adv}_A^P(\xi) = \left|2\Pr\left[\mathrm{Succ}_A^{GM_0}(\xi)\right] - 1\right|. \tag{1}$$

GM_1: In GM_1, Execute query is performed in GM_0. In other words, A obtains $\{M_1, M_2, M_3, M_4\}$. Since the values UN_i, CN_j, and TN_{CS} are unknown, A cannot obtain additional information to guess SK in Test query. Thus, we also have

$$\Pr[\mathrm{Succ}_A^{GM_1}(\xi)] = \Pr[\mathrm{Succ}_A^{GM_0}(\xi)]. \tag{2}$$

GM_2: In GM_2, Send query is performed in GM_1. According to Zipf's law [54], it implies

$$|\Pr[\mathrm{Succ}_A^{GM_2}(\xi)] - \Pr[\mathrm{Succ}_A^{GM_1}(\xi)]| \le q_{\mathrm{send}}/2^l. \tag{3}$$

GM_3: In GM_3, Hash query is performed in GM_2 without Send query. Based on the birthday paradox, it also implies

$$|\Pr[\mathrm{Succ}_A^{GM_3}(\xi)] - \Pr[\mathrm{Succ}_A^{GM_2}(\xi)]| \le q_{\mathrm{hash}}^2/2^{l+1}. \tag{4}$$

GM_4: In GM_4, Corrupt query is performed in GM_3. Two cases are set to analyze the security. The first case is to acquire long-term key or private value to simulate PFS and another case is to acquire the temporary value of some entity to simulate TVD attacks.

(1) Case I (PFS): A utilizes Π_{TCS}^z to acquire TCS's long-term key x or $\Pi_{S_j}^y$ to get the private value.
(2) Case II (TVD attacks): A utilizes $\Pi_{U_i}^x$, $\Pi_{S_j}^y$ or Π_{TCS}^z to get random number.

In Case I, $\{UN_i, CN_j, TN_{CS}\}$ are also unknown. A cannot obtain additional information to guess $SK = h(UN_i \oplus CN_j \oplus TN_{CS})$ in Test query. In Case II, even if A can obtain UN_i, $\{CN_j, TN_{CS}\}$ are also kept secret. Thus, we can obtain

$$|\Pr[\mathrm{Succ}_A^{\mathrm{GM}_4}(\xi)] - \Pr[\mathrm{Succ}_A^{\mathrm{GM}_3}(\xi)]| \leq q_{\mathrm{send}}/2^l + q_{\mathrm{hash}}^2/2^{l+1}. \tag{5}$$

GM_5: In GM_5, Corrupt query is also performed in GM_4 to simulate SSC attacks. In other words, A can access $\{UC_i, UE_i, UF_i, UG_i\}$. Similarly, with the help of the password dictionary, A attempts to guess the user's UPW_i. Because $\{UID_i, UPW_i\}$ are kept secret, A cannot calculate $UA_i = UPW_i \oplus h(BIO_i)$. Therefore, A cannot guess UPW_i. The probability of A guessing l bits of biometric information is $1/2^l$ [55]. According to Zipf's law [54], when $q_{\mathrm{send}} \leq 10^6$, the probability that A can guess the password is greater than 0.5. Thus, we can obtain

$$|\Pr[\mathrm{Succ}_A^{\mathrm{GM}_5}(\xi)] - \Pr[\mathrm{Succ}_A^{\mathrm{GM}_4}(\xi)]| \leq max\{C' \cdot q_{\mathrm{send}}^{s'}, q_{\mathrm{send}}/2^l\}. \tag{6}$$

In addition, the probability of A only correct guess c to win GM_5 is

$$\Pr[\mathrm{Succ}_A^{\mathrm{GM}_5}(\xi)] = 1/2. \tag{7}$$

Therefore, We can obtain the following results

$$\begin{aligned} Adv_A^P(\xi)/2 &= |\Pr[\mathrm{Succ}_A^{\mathrm{GM}_0}(\xi)] - 1/2| \\ &= |\Pr[\mathrm{Succ}_A^{\mathrm{GM}_0}(\xi)] - \Pr[\mathrm{Succ}_A^{\mathrm{GM}_5}(\xi)]| \\ &= |\Pr[\mathrm{Succ}_A^{\mathrm{GM}_1}(\xi)] - \Pr[\mathrm{Succ}_A^{\mathrm{GM}_5}(\xi)]| \\ &\leq \sum_{i=0}^{4} |\Pr[\mathrm{Succ}_A^{\mathrm{GM}_{i+1}}(\xi)] - \Pr[\mathrm{Succ}_A^{\mathrm{GM}_i}(\xi)]| \\ &= q_{\mathrm{send}}/2^{l-1} + q_{\mathrm{hash}}^2/2^l + max\{C' \cdot q_{\mathrm{send}}^{s'}, q_{\mathrm{send}}/2^l\} \end{aligned} \tag{8}$$

As a result, we can obtain

$$Adv_{\mathcal{A}}^{\mathcal{P}}(\xi) \leq q_{\mathrm{send}}/2^{l-2} + q_{\mathrm{hash}}^2/2^{l-1} + 2max\{C' \cdot q_{\mathrm{send}}^{s'}, q_{\mathrm{send}}/2^l\}. \tag{9}$$

□

5.2 ProVerif

ProVerif [56, 57] is an automated verification tool for analyzing and verifying authentication protocols. In this article, we utilize it to verify our protocol's security. It mainly simulates the "Registration" and "Login and authentication" process of U_i, S_j, and TCS by executing code.

The symbol definition that ProVerif needs to use is shown in Fig. 4a. In the query operation, we mainly query whether A can obtain the correct SK and whether the protocol is correct. The six events involved in our protocol are event UserStarted(), event UserAuthed(), event ControlServerAcUser(), event ControlServerAcCloudServer(), event CloudServerAcControlServer(), event UserAcControlServer(). They, respectively, indicate that U_i has started authentication, U_i has completed authentication,

```
(* channel*)
free ch:channel. (* public channel *)
free sch:channel [private]. (* secure channel, used for registering *)
(* shared keys *)
free SKi:bitstring [private].
free SKj:bitstring [private].
free SKcs:bitstring [private].
(* constants *)
free x:bitstring [private]. (* the CS's secret key *)
(* functions & reductions & equations *)
fun h(bitstring):bitstring. (*  hash function *)
fun mult(bitstring,bitstring):bitstring. (* scalar multiplication operation *)
fun add(bitstring,bitstring):bitstring. (* Addition operation *)
fun sub(bitstring,bitstring):bitstring. (* Subtraction operation *)
fun mod(bitstring,bitstring):bitstring. (* modulus operation *)
fun con(bitstring,bitstring):bitstring. (* concatenation operation *)
reduc forall m:bitstring,n:bitstring; getmess(con(m,n))=m.
fun xor(bitstring,bitstring):bitstring. (*  XOR operation *)
equation forall m:bitstring,n:bitstring; xor(xor(m,n),n)=m.
fun senc(bitstring,bitstring):bitstring. (* symmetric encryption *)
reduc forall m:bitstring,key:bitstring; sdec(senc(m,key),key)=m.
```

(a) Definitions

```
(* queries *)

query attacker(SKi).
query attacker(SKj).
query attacker(SKcs).
query inj-event(UserAuthed()) ==> inj-
event(UserStarted()).
query inj-event(ControlServerAcCloudServer())
==> inj-event(ControlServerAcUser()).
query inj-event(CloudServerAcControlServer())
==> inj-event(ControlServerAcCloudServer()).
query inj-event(UserAcControlServer()) ==>
inj-event(CloudServerAcControlServer()).

(* event *)
event UserStarted().
event UserAuthed().
event ControlServerAcUser().
event ControlServerAcCloudServer().
event CloudServerAcControlServer().
event UserAcControlServer().
```

(b) The queries and events

Fig. 4 Definitions, queries, and events

```
(* ----- Ui'   s process ----- *)
let ProcessUi=
new UIDi:bitstring;new UPWi:bitstring;
new BIOi:bitstring; new Ri:bitstring;
let RIDi=h(con(UIDi,Ri)) in out(sch,(UIDi,RIDi));
in(sch,(xTBi:bitstring,xti:bitstring));
let UAi=xor(UPWi,h(BIOi)) in let UCi=xor(Ri,UAi)
in let UEi=h(con(RIDi,UAi)) in
let UFi=xor(xTBi,UAi) in
let UGi=xor(xti,h(con(con(UIDi,UPWi),Ri))) in
!(event UserStarted();
 let UAi=xor(UPWi,h(BIOi)) in
let UEi'=h(con(UIDi,UAi)) in
if UEi'=UEi then new UNi:bitstring;
new TSi:bitstring;new CSIDj:bitstring;
let Ri=xor(UCi,UAi) in let RIDi=h(con(UIDi,Ri)) in
let xTBi=xor(UFi,UAi) in let
UHi=h(con(con(con(con(RIDi,CSIDj),UNi),TSi),xTB
i)) in let xti=xor(h(con(con(UIDi,UPWi),Ri)),UGi)
in let UJi=xor(h(con(xTBi,xti)),UNi) in
let UKi=xor(CSIDj,h(con(con(xTBi,UNi),xti))) in
out(ch,(UHi,UJi,UKi,RIDi,TSi));
(* ----- authentication ----- *)
event UserAuthed();
in(ch,(xTPcs:bitstring,xTQcs:bitstring));
let UYi=h(con(con(UNi,xTBi),CSIDj)) in
new CNj:bitstring; new TNcs:bitstring;
let Oi=xor(CNj,TNcs) in let Ti=xor(xTPcs,UYi)  in
let Oi=Ti in let SKi=h(con(con(UNi,CNj),TNcs)) in
let TQcs'=h(con(xor(CNj,TNcs),SKi)) in
if TQcs'= xTQcs then event UserAcControlServer();
 0
(* ----- authentication ----- *)).
```

(a) U_i process

```
(* ----- Sj process ----- *)
let ProcessSj=
new CSIDj:bitstring;new Rj:bitstring;
let RSIDj=h(con(CSIDj,Rj)) in
out (sch,(RSIDj,CSIDj));
in(sch,(yTSIDj:bitstring));
let CZj=xor(CSIDj,yTSIDj) in
!(
in(ch,(yUHi:bitstring,yUJi:bitstring,yUKi:bits
tring,yRIDi:bitstring,yTSi:bitstring));
new CNj:bitstring;new TSj:bitstring;
let yTSIDj=xor(CZj,CSIDj) in
let CLj=xor(yTSIDj,CNj)  in
let
CMj=h(con(con(con(con(con(CNj,CSIDj),yTS
IDj),yRIDi),yUHi),TSj)) in
out(ch,(yUHi,yUJi,yUKi,CLj,CMj,yRIDi,RSIDj,
yTSi,TSj));
in(ch,(yTPcs:bitstring,yTRcs:bitstring,yTQcs:
bitstring,yTVcs:bitstring));
let CWj=h(con(con(yTSIDj,CSIDj),CNj)) in
new UNi:bitstring;
new TNcs:bitstring;
let Pi=xor(UNi,TNcs) in
let Qi=xor(yTRcs,CWj) in
let Pi=Qi in
let SKj=h(con(con(UNi,CNj),TNcs)) in
let TVcs'=h(con(xor(UNi,TNcs),SKj)) in
if  TVcs'=yTVcs then event
CloudServerAcControlServer();
out(ch,(yTPcs,yTQcs));
0
).
```

(b) S_j process

```
(* ----- TCS process ----- *)
let UiReg = in(sch,(zUIDi:bitstring,zRIDi:bitstring));new
ti:bitstring;
let TBi=h(con(con(zRIDi,x),ti)) inout(sch,(TBi,ti));
0.(* -----Ui registration ----- *)
let SjReg =in (sch,(zRSIDj:bitstring,zCSIDj:bitstring));
let TSIDj=h(con(con(zRSIDj,zCSIDj),x)) in out(sch,(TSIDj));
0.(* -----Sj registration ----- *)
let TCSAuth =
in(ch,(zUHi:bitstring,zUJi:bitstring,zUKi:bitstring,zCLj:bitstrin
g,zCMj:bitstring,zRIDi:bitstring,zRSIDj:bitstring,zTSi:bitstring,
zTSj:bitstring));
new ti:bitstring;let TBi=h(con(con(zRIDi,x),ti)) in
let UNi=xor(zUJi,h(con(TBi,ti))) in
let CSIDj=xor(zUKi,h(con(con(TBi,UNi),ti))) in
let UHi'=h(con(con(con(con(zRIDi,CSIDj),UNi),zTSi),zTSj)) in
if UHi'= zUHi then event ControlServerAcUser();
(*-------TCS  verifies Ui----------*)
let TSIDj=h(con(con(zRSIDj,CSIDj),x)) in let
CNj=xor(h(TSIDj),zCLj) in
let CMj'=h(con(con(con(con(CNj,TSIDj),zRIDi),zUHi),zTSj)) in
if CMj'=zCMj  then  event ControlServerAcCloudServer();
(*-------TCS verifies Sj----------*)
new TNcs:bitstring;
let TPcs=xor(xor(CNj,TNcs),h(con(con(UNi,TBi),CSIDj))) in
let TRcs=xor(xor(UNi,TNcs),h(con(con(TSIDj,CSIDj),CNj))) in
let SKcs=h(xor(xor(UNi,CNj),TNcs)) in
let TQcs=h(con(xor(CNj,TNcs),SKcs)) in
let TVcs=h(con(xor(UNi,TNcs),SKcs)) in
out(ch,(TPcs,TRcs,TQcs,TVcs));0.
let ProcessTCS  = UiReg | SjReg | TCSAuth.
(* ----- Main ----- *)
process
(!ProcessUi | ! ProcessSj| !ProcessTCS)
```

(c) TCS process

Fig. 5 Execution process of U_i, S_j, TCS

TCS has successfully authenticated the U_i, TCS has successfully authenticated the S_j, S_j has successfully authenticated the TCS, and U_i has successfully authenticated the TCS. The specific queries and events required are shown in Fig. 4b.

The processes of using ProVerif to simulate U_i, S_j, and TCS are shown in Fig. 5. Here we take the ProVerif simulation TCS process as an example to describe the steps of the simulation process in detail. Among them, "UiReg" represents the U_i registration phase, "SjReg" represents the S_j registration phase, "TCSAuth" represents the TCS authentication phase, "new ti: bitstring" refers to the defined random number, "in(sch,(zUIDi: bitstring, zRIDi: bitstring))" refers to the TCS receives the registration request sent by the U_i, "out(ch,(TPcs, TRcs, TQcs, TVcs))" refers to the TCS

```
Verification summary:
Query not attacker(SKi[]) is true.
Query not attacker(SKj[]) is true.
Query not attacker(SKcs[]) is true.
Query inj-event(UserAuthed) ==> inj-event(UserStarted) is true.
Query inj-event(ControlServerAcCloudServer) ==> inj-event(ControlServerAcUser) is true.
Query inj-event(CloudServerAcControlServer) ==> inj-event(ControlServerAcCloudServer) is true.
Query inj-event(UserAcControlServer) ==> inj-event(CloudServerAcControlServer) is true.
```

Fig. 6 Verification result

sends messages to the S_j. The results of using the ProVerif authentication protocol are shown in Fig. 6. Experimental results are shown that our protocol is correct and can resist several well-known attacks, as well as each entity, can securely achieve authentication and establish SK.

5.3 Informal security analysis

5.3.1 Privileged insider attacks

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

5.3.2 TVD 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 SK is the same as above. Therefore, our protocol is secured against to temporary value disclosure (TVD) attacks.

5.3.3 User impersonation attacks

Suppose A intercept message $M_1 = \{UH_i, UJ_i, UK_i, RID_i, TS_i\}$, $M_2 = \{UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j\}$, $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$, $M_4 = \{TP_{CS}, TQ_{CS}\}$. Because A cannot get the private value $CSID_j$, UN_i and TB_i, so A cannot calculate the correct validation value UH_i, where $UH_i = h(RID_i \parallel CSID_j \parallel UN_i \parallel TB_i)$. Consequently, our protocol can withstand user impersonation attacks.

5.3.4 OPG attacks

Suppose A can get the data $\{UC_i, UE_i, UF_i, UG_i, h(.)\}$ stored in SC and try to guess the password UPW_i for U_i. Because A cannot obtain the biometric information BIO_i of U_i, A cannot correctly calculate UE_i, where $UE_i = h(UID_i \parallel UA_i)$ and $UA_i = UPW_i \oplus h(BIO_i)$. Consequently, our protocol can resist offline password guessing (OPG) attacks.

5.3.5 Replay attacks

Assume that A can intercept $\{M_1, M_2, M_3, M_4\}$ over the public channel. If A resends all the messages from the previous round, our protocol checks the validity of the current timestamp. If the timestamp is invalid, the session will terminate. Consequently, our protocol is immune to replay attacks.

5.3.6 Anonymity and untraceability

We use hash operation and random value to hide UID_i and $CSID_j$. During authentication, only the pseudo-identity RID_i or $RSID_j$ is used to ensure the anonymity of U_i and S_j. In addition, random numbers are differently used in each session, which also ensures that each entity is not traceable. Therefore, our protocol can ensure that entities are anonymity and untraceability.

5.4 Security and performance comparison

In this section, we will compare our protocol with five existing interrelated protocols [43, 45, 47–49] in terms of security and performance.

5.4.1 Security comparison

In terms of security comparison, the "✓" means that the protocol can resist such attacks, while the "×" means that it suffered from such attacks. In Table 4, it can be seen that Zhou et al.'s protocol [43] is insecure. Martinez-Pelaez et al.' protocol [45] is susceptible to reply attacks as well as cannot achieve mutual authentication and anonymity. Kang et al.'s protocol [47] is susceptible to OPG attacks. Huang et al.'s protocol [49] cannot withstand privileged insider and TVD attacks. Our protocol and Wu et al.'s protocol [48] can withstand all attacks.

5.4.2 Performance comparison

We only evaluate the computational and communication costs in the authentication and login phase. Our protocol performs only two operations: hash function and $\oplus$ operation. Due to the low cost of $\oplus$ and $||$ calculation, they are usually ignored in the computational process. In addition, we carried out simulation experiments to estimate the cost of the compared protocols. We use MI8 to simulate U_i, Lenovo laptops to simulate S_j, and Lenovo desktops to simulate TCS. The specific configuration and operation time of the equipment used in the simulation experiments are shown in Table 5, where the operation time is the average of 30 operations for each piece of equipment.

Table 4 The comparison of security

Security properties	[43]	[45]	[47]	[48]	[49]	Ours
Privileged insider attacks	×	✓	✓	✓	×	✓
TVD attacks	×	✓	✓	✓	×	✓
User impersonation attacks	×	✓	✓	✓	✓	✓
OPG attacks	✓	✓	×	✓	✓	✓
Replay attacks	✓	×	✓	✓	✓	✓
PFS	×	✓	✓	✓	✓	✓
Mutual authentication	×	×	✓	✓	✓	✓
Anonymity	✓	×	✓	✓	✓	✓
Untraceability	✓	✓	✓	✓	✓	✓

Table 5 Configuration and running time of equipment

	MI8	Lenovo laptop	Lenovo desktop
Operating system	Android system	Windows 10	Windows 10
Running memory	6G	8G	16G
CPU	Qualcomm snapdragon 845	Intel(R) Core(TM)i7-6700HQ CPU @ 2.60 GHz	Intel(R) Core(TM) i5-9500 CPU @ 3.00 GHz
Hash function	0.0041 ms	0.0034 ms	0.0023 ms
Symmetric encryption/decryption	0.2469 ms	0.1746 ms	0.1376 ms

Table 6 The comparisons of computational costs

protocol	U_i (ms)	S_j (ms)	*TCS* (ms)	Tocal (ms)
Zhou et al. [43]	$10T_H \approx 0.041$	$7T_H \approx 0.0238$	$19T_H \approx 0.0437$	0.1085
Martinez-Pelaez et al. [45]	$3T_E + 7T_H \approx 0.7694$	$3T_E + 6T_h \approx 0.5442$	$2T_E + 26T_H \approx 0.335$	1.6486
Kang et al. [47]	$8T_H \approx 0.0328$	$4T_H \approx 0.0136$	$11T_H \approx 0.0253$	0.0717
Wu et al. [48]	$12T_H \approx 0.0492$	$8T_H \approx 0.0272$	$19T_H \approx 0.0437$	0.1201
Huang et al. [49]	$8T_H \approx 0.0328$	$4T_H \approx 0.0136$	$10T_H \approx 0.023$	0.0694
Ours	$10T_H \approx 0.041$	$5T_H \approx 0.017$	$12T_H \approx 0.0276$	0.0856

Here, T_H and T_E are defined as the executing time of hash and symmetric encryption/decryption operations

Table 7 The comparisons of communicational costs

Protocols	Rounds	Cost (bits)
Zhou et al. [43]	4	4416
Martinez-Pelaez et al. [45]	6	4608
Kang et al. [47]	4	3712
Wu et al. [48]	5	4672
Huang et al. [49]	4	3392
Ours	4	3360

The computational cost results of different protocols are shown in Table 6. We can see that Martinez-Pelaez et al. [45] used symmetric encryption and decryption in the designed protocol, which has the highest computational cost. The computational cost of Wu et al.'s protocol [48] is lower than their protocol [45]. Since we add additional steps to enhance the security, the cost of our protocol is higher than both protocols [47] and [49].

In terms of communication cost, we assume that $|T|$, $|ID|$, $|R|$, $|H|$, $|E|$ are 32, 160, 160, 256, 256 bits, respectively. Four rounds of messages are transmitted in our protocol, $M_1 = \{UH_i, UJ_i, UK_i, RID_i, TS_i\}$, $M_2 = \{UH_i, UJ_i, UK_i, CL_j, CM_j, RID_i, RSID_j, TS_j\}$, $M_3 = \{TP_{CS}, TR_{CS}, TQ_{CS}, TV_{CS}\}$, $M_4 = \{TP_{CS}, TQ_{CS}\}$, where $\{UH_i, CM_j, TQ_{CS}, TV_{CS}\}$ belongs to hash value, $\{TS_i, TS_j\}$ belongs to timestamp, $\{RID_i, RSID_j\}$ belongs to identity, and $\{UJ_i, UK_i, CL_j, TP_{CS}, TR_{CS}\}$ belongs to random number. Thus, the communication cost of our protocol is $2|T| + 3|ID| + 8|R| + 6|H| = 3232$ bits. Similarly, the communication cost in [43] is $3|ID| + 15|R| + 6|H| = 4416$ bits, the communication cost in [45] is $3|T| + 3|ID| + 14|R| + |H| + 6|E| = 4608$ bits, the communication cost in [47] is $3|T| + 3|ID| + 10|R| + 6|H| = 3712$ bits, the communication

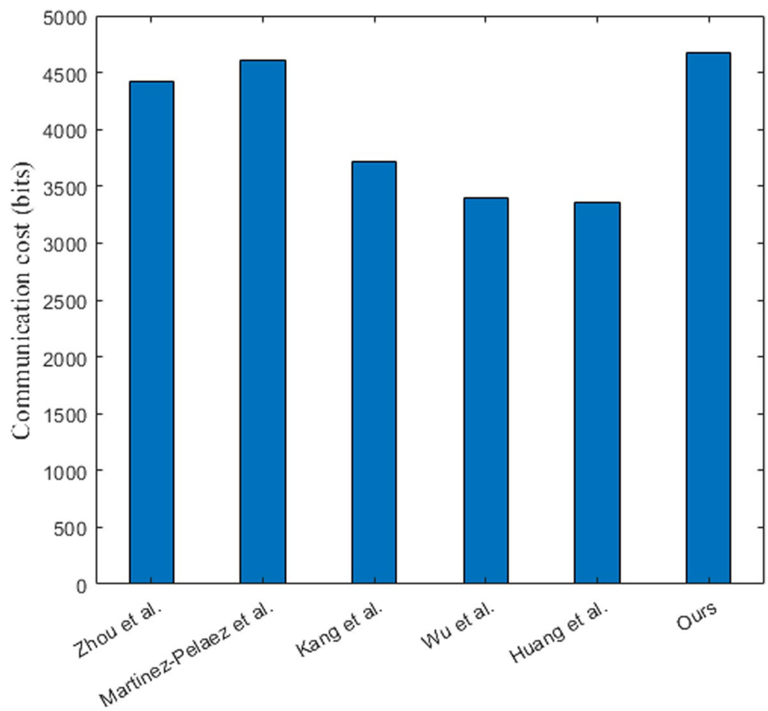

Fig. 7 Comparisons of communication cost

cost in [48] is $3|ID| + 15|R| + 7|H| = 4672$ bits, the communication cost in [49] is $3|T| + 3|ID| + 8|R| + 6|H| = 3392$ bits. The results are listed in Table 7 and depicted in Fig. 7. It is obvious that Wu et al.'s protocol [48] has the highest cost, followed by these protocols [43, 45, 47–49] and ours. In other words, our protocol has the lowest cost in communication.

In terms of security, we mainly list several common attacks, privileged insider, TVD, user impersonation, OPG, and replay attacks, and security requirements, PFS, mutual authentication, and anonymity. As shown in Table 4, we can see that only [48] and our protocol are secure. In another aspect, as shown in Tables 6, 7, and Fig. 7, the computational and communicational costs of [48] are higher than our protocol even if it is secure. Though the computational cost of our protocol is higher than the two protocols [47, 49]. However, they cannot resist some attacks. Overall, our protocol is the best authentication protocol with security and efficiency.

6 Conclusion

In this paper, we first summarize the importance of IoT-enabled devices in cloud environments and review the currently proposed AKA protocols related to these environments. Then, we have pointed Huang et al.'s protocol is insecure against privileged insider and TVD attacks. In order to solve the two security weaknesses, we have proposed an enhanced authentication protocol. Security analysis and comparisons are made to provide evidence to show our improvement is best currently.

Abbreviations

IoT Internet of Things

AKA	Authentication and key agreement
MITM	Man-in-the-middle
PUF	Physical unclonable functions
DoS	Denial of service
ECC	Elliptic curve cryptography
ABS	Attribute-based signature
IIoT	Industrial Internet of Things
TVD	Temporary value disclosure
SSC	Stolen smart card
REID	Radio frequency identification devices
PFS	Perfect forward secrecy
MA	Mutual authentication
SKD	Session key disclosure
OPG	Offline password guessing
ROR	Real or random.

Acknowledgments
Not applicable.

Author contributions
Conceptualization was contributed by C-MC and SK; methodology was contributed by T-YW and FK; software was contributed by QM; formal analysis was contributed by FK and SK; investigation was contributed by C-MC and QM; writing—original draft preparation, was contributed by T-YW and FK. All authors have read and agreed to the published version of the manuscript.

Funding
This research was partially supported by Natural Science Foundation of Shandong Province, China (Grant no. ZR202111230202)

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 16 April 2022 Accepted: 21 April 2023
Published online: 27 April 2023

References

1. S. Das, S. Namasudra, Macpabe: multi-authority-based cp-abe with efficient attribute revocation for iot-enabled healthcare infrastructure. Int. J. Netw. Manag. (2022). https://doi.org/10.1002/NEM.2200
2. X. Li, S. Liu, S. Kumari, C.-M. Chen, Psap-wsn: a provably secure authentication protocol for 5g-based wireless sensor networks. CMES-Comput. Model. Eng. Sci. **135**(1), 711–732 (2023)
3. S. Das, S. Namasudra, Multi-authority cp-abe-based access control model for iot-enabled healthcare infrastructure. IEEE Trans. Ind. Inf. **19**(1), 821–829 (2023)
4. X. Xue, C. Jiang, Matching sensor ontologies with multi-context similarity measure and parallel compact differential evolution algorithm. IEEE Sens. J. **21**(21), 24570–24578 (2021)
5. X. Xue, Q. Huang, Generative adversarial learning for optimizing ontology alignment. Expert Syst. (2022). https://doi.org/10.1111/exsy.12936
6. J.S. Pan, B. Sun, S.C. Song, M. Chu, C.S. Zhu, A. Shieh, Parallel compact gannet optimization algorithm for solving engineering optimization problems. Mathematics **11**(2), 439 (2023)
7. S. Chaudhry, Combating identity de-synchronization: an improved lightweight symmetric key based authentication scheme for iov. J. Netw. Intell. **6**, 656–667 (2021)
8. H. Xiong, J. Chen, Q. Mei, Y. Zhao, Conditional privacy-preserving authentication protocol with dynamic membership updating for vanets. IEEE Trans. Dependable Secure Comput. **19**(3), 2089–2104 (2022)
9. J.-S. Pan, J.-X. Lv, L.-J. Yan, S.-W. Weng, S.-C. Chu, J.-K. Xue, Golden eagle optimizer with double learning strategies for 3d path planning of uav in power inspection. Math. Comput. Simul. **193**, 509–532 (2022)
10. M.B. Mollah, J. Zhao, D. Niyato, K.-Y. Lam, X. Zhang, A.M. Ghias, L.H. Koh, L. Yang, Blockchain for future smart grid: A comprehensive survey. IEEE Internet Things J. **8**(1), 18–43 (2020)
11. Y. Luo, W. Zheng, Y.-C. Chen, An anonymous authentication and key exchange protocol in smart grid. J. Netw. Intell. **6**(2), 206–215 (2021)
12. B. Pradhan, S. Bhattacharyya, K. Pal, Iot-based applications in healthcare devices. J. Healthc. Eng. **2021**, 6632599 (2021)
13. T.-Y. Wu, Q. Meng, L. Yang, S. Kumari, M. Pirouz, Amassing the security: An enhanced authentication and key agreementprotocol for remote surgery in healthcare environment. CMES-Comput. Model. Eng. Sci. **134**(1), 317–341 (2023)

14. P. Bedi, S. Das, S. Goyal, P.K. Shukla, S. Mirjalili, M. Kumar, A novel routing protocol based on grey wolf optimization and q learning for wireless body area network. Expert Syst. Appl. **210**, 118477 (2022)
15. A. Yassine, S. Singh, M.S. Hossain, G. Muhammad, Iot big data analytics for smart homes with fog and cloud computing. Future Gener. Comput. Syst. **91**, 563–573 (2019)
16. T.-Y. Wu, F. Kong, L. Wang, Y.-C. Chen, S. Kumari, J.-S. Pan, Toward smart home authentication using puf and edge-computing paradigm. Sensors **22**(23), 9174 (2022)
17. P. Chithaluru, F. Al-Turjman, M. Kumar, T. Stephan, Mtcee-lln: Multilayer threshold cluster-based energy-efficient low-power and lossy networks for industrial internet of things. IEEE Internet Things J. **9**(7), 4940–4948 (2021)
18. S. Chandra, W. Yafeng, Cloud things construction-the integration of internet of things and cloud computing. Future Gener. Comput. Syst. **56**(C), 684–700 (2016)
19. M.F. Mushtaq, U. Akram, I. Khan, S.N. Khan, A. Shahzad, A. Ullah, Cloud computing environment and security challenges: A review. Int. J. Adv. Comput. Sci. Appl. **8**(10), 183–195 (2017)
20. T.-Y. Wu, Q. Meng, L. Yang, X. Guo, S. Kumari, A provably secure lightweight authentication protocol in mobile edge computing environments. J. Supercomput. **78**, 13893–13914 (2022)
21. M. Turkanović, B. Brumen, M. Hölbl, 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. **20**, 96–112 (2014)
22. M.S. Farash, M. Turkanović, S. Kumari, M. Hölbl, An efficient user authentication and key agreement scheme for heterogeneous wireless sensor network tailored for the internet of things environment. Ad Hoc Netw. **36**, 152–176 (2016)
23. R. Amin, S.H. Islam, G. Biswas, M.K. Khan, L. Leng, N. Kumar, Design of an anonymity-preserving three-factor authenticated key exchange protocol for wireless sensor networks. Comput. Netw. **101**, 42–62 (2016)
24. U. Chatterjee, R.S. Chakraborty, D. Mukhopadhyay, A puf-based secure communication protocol for iot. ACM Trans. Embed. Comput. Syst. (TECS) **16**(3), 1–25 (2017)
25. A. Braeken, Puf based authentication protocol for iot. Symmetry **10**(8), 352 (2018)
26. P.K. Panda, S. Chattopadhyay, A secure mutual authentication protocol for iot environment. J. Reliab. Intell. Environ. **6**(2), 79–94 (2020)
27. Y. Bao, W. Qiu, X. Cheng, Efficient and fine-grained signature for iiot with resistance to key exposure. IEEE Internet Things J. **8**(11), 9189–9205 (2021)
28. P. Chithaluru, T. Stephan, M. Kumar, A. Nayyar, An enhanced energy-efficient fuzzy-based cognitive radio scheme for iot. Neural Comput. Appl. **34**(21), 19193–19215 (2022)
29. H. Liu, H. Ning, Q. Xiong, L.T. Yang, Shared authority based privacy-preserving authentication protocol in cloud computing. IEEE Trans. Parallel Distrib. Syst. **26**(1), 241–251 (2014)
30. S. Kalra, S.K. Sood, Secure authentication scheme for iot and cloud servers. Pervasive Mob. Comput. **24**, 210–223 (2015)
31. R. Amin, N. Kumar, G. Biswas, R. Iqbal, V. Chang, A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment. Future Gener. Comput. Syst. **78**, 1005–1019 (2018)
32. K. Xue, P. Hong, C. Ma, A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture. J. Comput. Syst. Sci. **80**(1), 195–206 (2014)
33. M.-C. Chuang, M.C. Chen, An anonymous multi-server authenticated key agreement scheme based on trust computing using smart cards and biometrics. Expert Syst. Appl. **41**(4), 1411–1418 (2014)
34. C. Wang, K. Ding, B. Li, Y. Zhao, G. Xu, Y. Guo, P. Wang, An enhanced user authentication protocol based on elliptic curve cryptosystem in cloud computing environment. Wirel. Commun. Mob. Comput. **2018**, 3048697 (2018)
35. F. Wu, L. Xu, X. Li, A new chaotic map-based authentication and key agreement scheme with user anonymity for multi-server environment. In: International Conference on Frontier Computing, pp. 335–344 (2017). Springer
36. D. Wang, X. Zhang, Z. Zhang, P. Wang, Understanding security failures of multi-factor authentication schemes for multi-server environments. Comput. Secur. **88**, 101619 (2020)
37. K. Fan, Q. Luo, K. Zhang, Y. Yang, Cloud-based lightweight secure rfid mutual authentication protocol in iot. Inf. Sci. **527**, 329–340 (2020)
38. D. He, S. Zeadally, N. Kumar, W. Wu, Efficient and anonymous mobile user authentication protocol using self-certified public key cryptography for multi-server architectures. IEEE Trans. Inf. Forensics Secur. **11**(9), 2052–2064 (2016)
39. Y. Yu, L. Hu, J. Chu, A secure authentication and key agreement scheme for iot-based cloud computing environment. Symmetry **12**(1), 150 (2020)
40. A. Irshad, S.A. Chaudhry, O.A. Alomari, K. Yahya, N. Kumar, A novel pairing-free lightweight authentication protocol for mobile cloud computing framework. IEEE Syst. J. **15**(3), 3664–3672 (2020)
41. D. Rangwani, H. Om, A secure user authentication protocol based on ecc for cloud computing environment. Arab. J. Sci. Eng. **46**(4), 3865–3888 (2021)
42. M. Wazid, A.K. Das, S. Kumari, X. Li, F. Wu, Provably secure biometric-based user authentication and key agreement scheme in cloud computing. Secur. Commun. Netw. **9**(17), 4103–4119 (2016)
43. L. Zhou, X. Li, K.-H. Yeh, C. Su, W. Chiu, Lightweight iot-based authentication scheme in cloud computing circumstance. Future Gener. Comput. Syst. **91**, 244–251 (2019)
44. F. Wang, G. Xu, G. Xu, Y. Wang, J. Peng, A robust iot-based three-factor authentication scheme for cloud computing resistant to session key exposure. Wirel. Commun. Mob. Comput. **2020**, 1–5 (2020)
45. R. Martínez-Peláez, H. Toral-Cruz, J.R. Parra-Michel, V. García, L.J. Mena, V.G. Félix, A. Ochoa-Brust, An enhanced lightweight iot-based authentication scheme in cloud computing circumstances. Sensors **19**(9), 2098 (2019)
46. S. Yu, K. Park, Y. Park, A secure lightweight three-factor authentication scheme for iot in cloud computing environment. Sensors **19**(16), 3598 (2019)
47. B. Kang, Y. Han, K. Qian, J. Du, Analysis and improvement on an authentication protocol for iot-enabled devices in distributed cloud computing environment. Math. Probl. Eng. **2020**, 3048697 (2020)
48. H.-L. Wu, C.-C. Chang, Y.-Z. Zheng, L.-S. Chen, C.-C. Chen, A secure iot-based authentication system in cloud computing environment. Sensors **20**(19), 5604 (2020)

49. H. Huang, S. Lu, Z. Wu, Q. Wei, An efficient authentication and key agreement protocol for iot-enabled devices in distributed cloud computing architecture. EURASIP J. Wirel. Commun. Netw. **2021**(1), 1–21 (2021)
50. D. Dolev, A. Yao, On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)
51. H. Canetti, Ranand Krawczyk: analysis of key-exchange protocols and their use for building secure channels, in *Adv. Cryptol. EUROCRYPT 2001*. ed. by B. Pfitzmann (Springer, Berlin, Heidelberg, 2001), pp.453–474
52. R. Canetti, O. Goldreich, S. Halevi, The random oracle methodology, revisited. J. ACM (JACM) **51**(4), 557–594 (2004)
53. T. Wu, X. Guo, Y. Chen, S. Kumari, C. Chen, Amassing the security: an enhanced authentication protocol for drone communications over 5g networks. Drones **6**(1), 10 (2022)
54. D. Wang, H. Cheng, P. Wang, X. Huang, G. Jian, Zipf'slaw in passwords. IEEE Trans. Inf. Forensics Secur. **12**(11), 2776–2791 (2017)
55. V. Odelu, A.K. Das, A. Goswami, A secure biometrics-based multi-server authentication protocol using smart cards. IEEE Trans. Inf. Forensics Secur. **10**(9), 1953–1966 (2015)
56. B. Blanchet et al., An efficient cryptographic protocol verifier based on prolog rules. Csfw **1**, 82–96 (2001)
57. T.-Y. Wu, L. Wang, X. Guo, Y.-C. Chen, S.-C. Chu, Sakap: Sgx-based authentication key agreement protocol in iot-enabled cloud computing. Sustainability **14**(17), 11054 (2022)

Publisher's Note