Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **8**, Number **3**, August **2023**

DACSC: Secure Authentication Protocol Based on Dynamic Authentication Credentials and IntelSGX in Cloud Computing Environments

Lina Ni, BoGuang Ni, Yuncan Tang, Jinquan Zhang*

College of Computer Science and Engineering
Shandong University of Science and Technology
Qingdao, 266590, China
nln2004@163.com, nihualong1@163.com, tyc18265475128@163.com, tjzhangjinquan@126.com

*Corresponding author: Jinquan Zhang

Received March 6, 2023, revised May 17, 2023, accepted June 22, 2023.

Abstract. *Cloud computing is a new super computing paradigm, which has greatly changed the way users store and process data. Meanwhile, with the prosperity of cloud computing, there are many hidden dangers in data privacy and security. Many secure authentication protocols in cloud computing environment have been proposed. However, most of the authentication schemes are vulnerable to various attacks. Therefore, it is vital to plan a secure and effective authentication protocol in the cloud computing surroundings. In this paper, we propose a secure authentication protocol DACSC based on dynamic authentication credential (DAC) and Intel software guard extensions (SGX) for the cloud computing environment. In order to prevent internal attacks and information leakage stored in memory, we store key data in the SGX. Furthermore, we adopt dynamic identity authentication credentials to timely update the server's pseudo identity information. We conduct a formal safety analysis of our DACSC protocol adopting ProVerif which is a formal security evidence tool under the stochastic prediction model. Moreover, We also conduct other analysis to prove that DACSC can resist internal attacks, simulation attacks, and achieve user anonymity. By comparing security and performance, we concluded that DACSC is relatively secure and protects the privacy of data.*
Keywords: Cloud computing, Dynamic authentication credential, SGX, Authentication

1. **Introduction.** Recently, IoT [1, 2] and cloud computing technology have developed rapidly. People could use the resources in the cloud server at anytime and anywhere [3], it has also been widely used in medical services [4,5]. When using cloud services, not only do you need to improve the efficiency of the transfer [6], but you also need to protect the security of the data [7]. For example, in a typical scenario of cloud computing, users want to store some files, videos, audio and other information to the cloud server through mobile phones, computers and other Internet of things devices, and the cloud server will provide these services. The control server is a trusted third party. They all need to register with the control server to obtain legitimacy, and then the three parties establish legal session keys to ensure the safety of data during transmission.

In cloud computing environments [8–12], the information transmission between the user and the server is conducted in an insecure channel, which is vulnerable to various attacks from opponents, resulting in some information that the user does not want to expose, such as mobile phone number and ID number, being stolen by illegal elements. Several

scholars have proposed authentication protocols for this environment [10–13], however, these schemes con't prevent the leakage of information. When the opponent attacks many times, it is easy to be broken by the opponent because some key parameters cannot be updated in time, so the three parties will not know whether their identity information is captured by the adversary after many communications, and if the identity information is not updated in time, it will lead to very serious consequences [14]. Moreover, common authentication protocols usually store authentication information and keys in memory, which is vulnerable to internal attacks by privileged users, and once an adversary obtains this information, it can guess what we want to protect based on the known information. Therefore, a trusted execution environment for storing identity information and keys is essential.

We combine Intel software guard extensions (SGX) with dynamic authentication credentials (DAC) to improve the security of the protocol. SGX provides security and integrity for the data stored in it [15]. We store random numbers and the master key. Since this information is not available to the adversary, our protocol can prevent information from being obtained by privileged users. Thus, adversaries can't steal it by pretending to be legal identities to meet the requirements. In the process of information transmission, even if the attacker does not know the information we protect at first, if the information is transmitted back and forth many times, it can be easily guessed, so we dynamically update the authentication credentials to ensure that the identity is not be guessed.

The main contributions of this paper are as follows:

1. We propose an authentication protocol DACSC based on SGX protocol that stores key parameters and random numbers in a confidential environment during information transmission, which can effectively protect users' private data.
2. We present an authentication credential mechanism that can effectively update identity information and prevent adversary attacks in the case of multiple rounds of information transmission.

2. **Related Work.** In cloud computing environments, in order to protect the security of information, many experts have done some research, which we review in this section.

Xue et al. [16] proposed a scheme for dynamic authentication of pseudonym identity and multi-server architecture. Amin et al. [11] proved that their protocol cannot guarantee that the user's identity will not be disclosed. They also presented an authentication that can be used in distributed cloud environments without security attacks protocol to eliminate the security issues associated with Xue et al. [16]. Unfortunately, Challa et al. [17] confirmed that internal and simulated attacks could not be resisted.

Tsai et al. [18] still advanced a distributed mobile cloud computing service based authentication protocol that uses bilinear pairing to guarantee anonymity. However, He et al. [19] turned out that their protocol is not resistant to simulation attacks and proposed a new protocol that uses an identity-based signature scheme.

Zhou et al. [12] presented an authentication protocol for cloud servers based on IoT architecture using lightweight encryption and claimed to be resistant to multiple attacks. However, Pelaez et al. [13] proved that it cannot resist insider attacks, offline password guessing attacks, and cannot guarantee the security of session keys, and proposed an improved scheme based on Zhou et al. [12] to close the security gap and achieve user anonymity. In the same year, Yu et al. [10] found that the protocol of Pelaez et al. [13] was not secure against simulation attacks, replay attacks, insecure session keys, and could not achieve user anonymity. To address these security issues, Yu et al. [10] proposed a lightweight authentication protocol in which there is no dynamic update credential phase and security is not guaranteed in their scheme. Kang et al. [20] proposed a protocol for

Table 1. Notations and their meanings

Notations	Meanings
U_i, S_j	Registered users, registered cloud servers
CS	Control Server
SID_j	Identity of the cloud server
ID_i	The identity of the user
PW_i	User's password
BIO_i	User's biometric information
TID_i	User's Pseudo-identity
$PSID_j$	Cloud server's Pseudo-identity
HPW_i	Pseudo-passwords for user
x_{GW}	Master key for the control server
T	Timestamp
$h(\cdot)$	Single hash function
$x\|\|y$	Concatenation
$\oplus$	XOR operation

IoT equipments, however, Li et al. [21] demonstrated that their scheme is susceptible to session-specific of ad hoc message attacks.

Many scholars have worked on SGX technology, Balisane and Martin [22] proposed proposes to use trusted execution environment to store data for authentication that addresses some problems and act against some of the danger posed by the presence of malware. Condé et al. [23] used Intel SGX on UNIX system to protect authentication credentials, proposed a cryptographic file protection scheme.

The above methods are slightly inadequate in terms of security and high in terms of communication cost. However, they cannot well meet the needs in the cloud computing environment. To solve the above problems, we propose a secure authentication protocol based on dynamic authentication credentials and IntelSGX to further improve security and reduce the communication cost.

3. Preparation and system model.

In this section, we depict the architecture of our proposed DACSC system model and the main idea of Intel SGX.

The symbols used in this paper are described in Table 1.

3.1. System model.

The system model of our DACSC protocol in the cloud computing environment is shown in Figure 1. There are three participants in the model: user, cloud server, and control server. First, the subscriber and the cloud server first register with the control server, which passes the registered information back over a common channel and then stores a portion of the information in the SGX. After registration, the legitimate user and server log in and authenticate, eventually negotiating a common session key to enable secure communication between the three parties. The details of each participant are described as follows.

(1) User: Users store information in the cloud through the Internet of things such as mobile phones and computers, and the server provides services for them. They share session keys. In our DACSC model, the user registers with the control server and stores the registration information in the trusted execution environment.

(2) Cloud server: It is semi-honest and can provide relevant services. It may attack the information we transmit, but does not leak the information to adversaries or help them

to attack. In the DACSC model, it also needs to store and call private information in SGX.

(3) Control server: It is also semi-honest and is a registration center. In the DASC model, after receiving the registration information, call the information in SGX to confirm its legal identity through identification and verification, and update the authentication credentials dynamically.

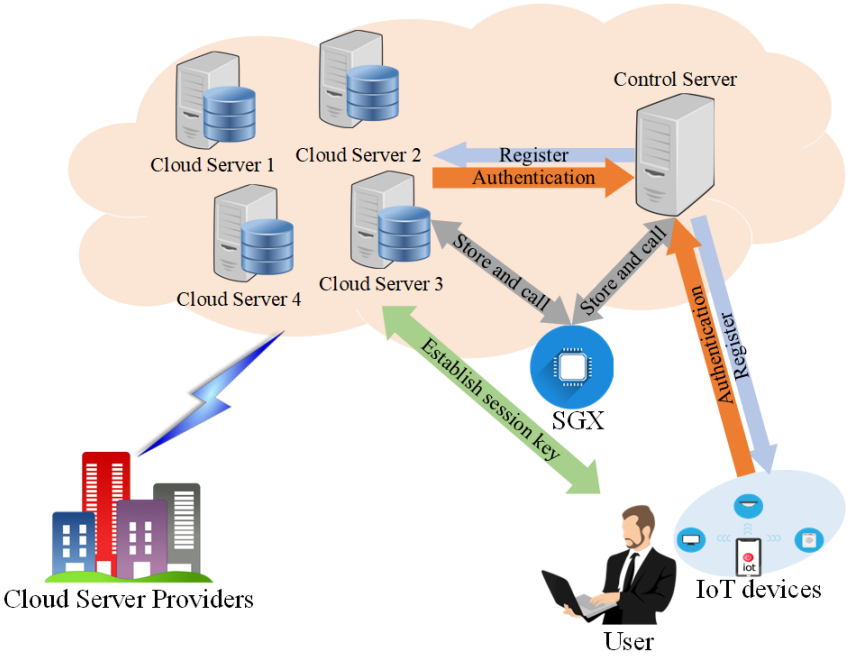

FIGURE 1. DACSC System Model.

3.2. **Intel SGX.** Intel SGX [24], introduced by Intel, is a set of CPU instruction annexe that provide security to the hardware and can effectively protect the code and data stored in it. SGX calculates in the CPU to avoid some attacks from the system [23,25–27]. PRM is a random area reserved in dynamic memory, this contiguous space cannot be accessed by any software. The Enclave Page Cache (EPC) is a set of memory reserved in advance. By running code, you can get some sensitive information and data from the safe area. These information and data are always encrypted, which is very secure, and it is impossible for opponents to obtain them.

SGX will divide two regions. The untrusted zone can only call the content of the trusted zone through Ecall code, and the trusted zone can only call the content of the untrusted zone through Ocall code. In our protocol, identity information, random numbers, master keys and other data are stored in SGX through Ecall code at appropriate times, and can be obtained from it through Ocall code when necessary. Because the memory area in SGX is confidential and cannot be broken by opponents, the security of our protocol is guaranteed.

In our DACSC protocol, we use SGX to store pseudo-identities and random numbers during the registration phase, and call the stored data during the login and authentication phases to enhance the security of DACSC.

4. **Proposed protocol.** In this section, we present the DACSC protocol. Our protocol includes three phases. It is noting that the authentication phase is the core of our DACSC protocol. In this phase, we implement dynamic identity credential update. In addition, we access the information under the protection of SGX's trusted execution environment. The details of DACSC protocol are described as follows.

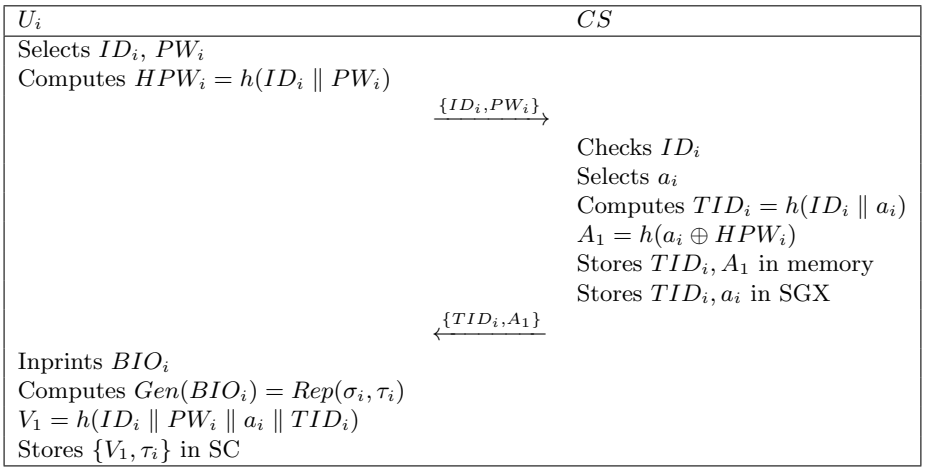

FIGURE 2. User registration phase

4.1. **User registration phase.** When a user wants to stump in to the server, he/she must register with the control server, going through the following steps. We describe this phase in Figure 2.

1. User first enters its identity and password, calculates $HPW_i = h(ID_i \parallel PW_i)$, and then broadcast the registration information ID_i, PW_i to the control server CS through the common channel.

   When CS receives the registration information from the user, it first checks whether the user's identity has been registered, and if not, generates a random number a_i and calculates $TID_i = h(ID_i \parallel a_i)$, $A_1 = h(a_i \oplus HPW_i)$, then stores the parameter $\{TID_i, A_1\}$ in the smart card SC, stores $\{TID_i, a_i\}$ in the hardware SGX, and returns$\{TID_i, A_1\}$ to the user.
2. When the user receives the message back from the control server CS, the user's biometric BIO_i is scanned and the generation algorithm calculates $Gen(BIO_i) = Rep(\sigma_i, \tau_i)$, $V_1 = h(ID_i \parallel PW_i \parallel a_i \parallel TID_i)$, the limit $\{V_1, \tau_i\}$ are then store in memory.

4.2. **Cloud server registration phase.** Before provide services to legitimate users, it also needs to register with the control server, and we describe this phase in Figure 3, with the following process.

1. The cloud server choose a name and broadcast it over a secure channel to the CS.
2. After the server receives the data sent by S_j, it first checks whether SID_j already exists in the database, if so, cancels the registration request. Otherwise CS generates a random number b_j and calculates $PSID_j = h(SID_j \parallel b_j)$, $K_{SG} = h(SID_j \parallel x_{GW})$ and then stores $\{PSID_j, b_j\}$ in memory, stores $\{PSID_j, K_{SG}\}$ in hardware SGX, and sends $\{PSID_j, K_{SG}, SID_j\}$ back to the cloud server via secure channel, which stores $\{PSID_j\}$ in memory and stores $\{PSID_j, K_{SG}, SID_j\}$ in hardware SGX. The cloud server registration phase is completed.

S_j	CS
	Selects SID_j, b_j Computes $PSID_j = h(SID_j \parallel b_j)$ $K_{SG} = h(SID_j \parallel x_{GW})$ Stores $\{PSID_j, b_j\}$ in memory Stores $\{PSID_j, K_{SG}\}$ in SGX
$\xleftarrow{\{PSID_j, K_{SG}, SID_j\}}$	
Stores $\{PSID_j\}$ in memory Stores $\{PSID_j, K_{SG}, SID_j\}$ in SGX	

FIGURE 3. Cloud server registration phase

4.3. **Login and authentication phase.** The comprehensive steps of this stage are as follows, and the process is shown in Figure 4.

1. When a legitimate user U_i wants to use the service provided by cloud server node S_j to communicate securely with it, he first needs to insert the user's smart card, enter the user name ID_i and password PW_i, scan his biometric information BIO_i, and then calculate σ_i through the regenerative function $Rep(BIO_i, \tau_i)$ of the fuzzy extractor, then calculate $V_1' = h(ID_i \parallel PW_i \parallel \sigma_i \parallel TID_i)$, next, check whether V_1' is equal to V_1 and if so, pass the memory verification. Next, the user generates random number n_1, timestamp T_1, computes $HPW_i = h(ID_i \parallel PW_i)$, $a_i = A_1 \oplus HPW_i$, $A_2 = n_1 \oplus h(a_i \parallel TID_i)$, $A_3 = SID_j \oplus h(HPW_i \parallel T_1)$, $V_2 = h(a_i \parallel n_1 \parallel SID_j \parallel T_1)$, and finally the user sends the message $\{ A_2, A_3, V_2, T_1, TID_i\}$.
2. After get the message from the user, the cloud server S_j produce chance number n_2 and timestamp T_2 then sends the pseudo-identity $PSID_j$ of the cloud server, and calls the information stored in it SID_j and K_{SG} through the security interface, then calculates $A_4 = n_2 \oplus h(SID_j \parallel K_{SG})$, $V_3 = h(n_2 \parallel SID_j \parallel T_2)$ and finally sends $M_2 = \{A_2, A_3, V_2, T_1, TID_i, A_4, V_3, T_2, PSID_j\}$ to the control server.
3. CS uses A_1 to retrieve the user's pseudo-identity TID_i in the database, then sends the user's pseudo-identity TID_i to the security interface of SGX, calls the information stored in it through the security interface, then calculates $n_1 = A_2 \oplus h(a_i \parallel TID_i)$, $SID_j = A_3 \oplus h(HPW_i \parallel T_1)$, $V_2' = h(a_i \parallel n_1 \parallel SID_j \parallel T_1)$, and verify whether the calculated value V_2' is equal to the received value V_2. If it is not equal, the current session is terminated. Instead, the control server sends the pseudo-identity TID_i of the cloud server to the security interface of SGX, calls the information K_{SG} stored in it through the security interface, and then calculates $n_2 = A_4 \oplus h(SID_j \parallel K_{SG})$, $V_3' = h(n_2 \parallel SID_j \parallel T_2)$, and verify whether the calculated value V_3' is equal to the received value V_3. If it is not equal, the current session is terminated. Instead, generate a random number n_3, and compute $SK = h(n_1 \oplus n_2 \oplus n_3 \oplus SID_j)$, $PSID_j^{new} = h(SID_j \parallel n_1)$, $PSID_j^{new'} = PSID_j^{new} \oplus h(SID_j \parallel n_1)$, $TID_i^{new} = h(TID_i \parallel n_1 \parallel a_i)$, $TID_i^{new'} = TID_i^{new} \oplus h(n_1 \parallel a_i)$, $A_5 = (n_1 \oplus n_3) \oplus h(n_2 \parallel SID_j \parallel K_{SG})$, $A_6 = (n_2 \oplus n_3) \oplus h(SID_j \parallel n_1)$, $V_4 = h(SK \parallel n_1 \oplus n_3 \parallel PSID_j^{new})$, $V_5 = h(SK \parallel n_2 \oplus n_3 \parallel TID_i^{new})$.Finally the control server transmits the messages $\{PSID_j^{new'}, TID_i^{new'}, A_5, A_6, V_4, V_5\}$ back to the cloud server.
4. When the cloud server receives the message back from the control server, it calculates $PSID_j^{new} = PSID_j^{new'} \oplus h(SID_j \parallel n_1)$, $(n_1 \oplus n_3) = A_5 \oplus h(n_2 \parallel SID_j \parallel K_{SG})$, $SK = h(n_1 \oplus n_2 \oplus n_3 \oplus SID_j)$, $V_4' = h(SK \parallel n_1 \oplus n_3 \parallel PSID_j^{new})$, and verify whether the computed value V_4' is equal to the received value V_4. If it is not equal, the current session is terminated. Instead, the cloud server replace the $PSID_j^{new}$ with $PSID_j$ in memory. Finally, the cloud server sends the message $\{ TID_i^{new'}, A_6, V_5\}$ to the user.

FIGURE 4. Login and authentication phase

5. When the user receives the information from the cloud server, $TID_i^{new} = TID_i^{new'} \oplus h(n_1 \parallel a_i)$, $(n_2 \oplus n_3) = A_6 \oplus h(SID_j \parallel n_1)$, $SK = h(n_1 \oplus n_2 \oplus n_3 \oplus SID_j)$, $V_5' = h(SK \parallel n_2 \oplus n_3 \parallel TID_i^{new})$, and verify whether the computed value V_5' is equal to the received value V_5. If it is not equal, the current session is terminated. Instead, the cloud server and the user achieves mutual authentication and negotiates a temporary session key, and finally store the new pseudo-identities $TID_i^{new}, PSID_j^{new}$ of the user and the cloud server into the smart card.

5. **Security analysis.**

5.1. **Formal security analysis.** In this section, a formal safety analysis is conducted to prove that the code of conduct is safe and accurate.

The C-K model [28] is an expansion of the D-Y model [29], and under the assumptions of the C-K model [28], the following capabilities of A are defined.

1. Suppose that after its attack, the message may not be able to spread, or it may be the modified information, and it can also eavesdrop on the message on the public channel.
2. A can obtain the data stored by users in SC in some way.
3. A can be an internal person who can access the information stored in the control server.
4. A can guess the user's relevant information, but Acannot guess the identity and password at the same time in a certain time.

5.1.1. *Main proofs based on ROR model.* We turn out the safety of the proposed protocol using Random Oracle(ROR)model [30, 31]. The protocol consists of three entities. In this model, $\prod_{U_i}^{x}$, $\prod_{S_j}^{y}$ and $\prod_{CS}^{z}$ denote the x-th user, the y-th cloud server, and the z-th control server, respectively. Suppose that the query function of adversary $\mathcal{A}$ consists of the following three: $Z = \prod_{U_i}^{x}, \prod_{S_j}^{y}, \prod_{CS}^{z}$.

$Execute(Z)$: This query simulates the ability of an attacker $\mathcal{A}$ to perform a passive attack. $\mathcal{A}$ executes this query to intercept messages transmitted over the public channel.

$Send(Z, I)$: This query simulates the ability of attacker $\mathcal{A}$ to perform an active attack, assuming that $\mathcal{A}$ executes this query and he sends message I to Z and receives a reply from Z.

$Hash(string)$: The attacker $\mathcal{A}$ can query and return the hash value through a string.

$Corrupt(Z)$: Assuming that the attacker $\mathcal{A}$ can obtain private information after performing this operation.

$Test(Z)$: Suppose that the attacker $\mathcal{A}$ performs an operation equivalent to a coin toss. The number obtained can be 1 or 0. If the number obtained is 0, the correct SK can be obtained. Otherwise, it gets other values of the same length.

Theorem. In the ROR model, if $\mathcal{A}$ an execute the above five queries, the probability of successful attack $\mathcal{P}$ is $Adv_{\mathcal{A}}^{\mathcal{P}}(\xi) \leq q_{send}/2^{l-2} + 3q_{hash}^2/2^{l-1} + 2max\{C', q_{send}^{s'}, q_{send}/2^l\}$, where q_{send} refers to execution times, q_{hash} is the time to execute hash function, C' and s' are two constants, and l is the bit length of the biological information.

Proof. Assume there are five games $GM_0, GM_1, GM_2, GM_3, GM_4, Succ_{\mathcal{A}}^{GM_i}(\xi)$ is the probability that $\mathcal{A}$ can win GM_0 to GM_4.

Game GM_0. GM_0 denotes the first round, starting with an unbiased coin toss. The advantage of $\mathcal{A}$ is:

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

Game GM_1. This game executes $Execute(Z)$, it simulates stealing information. At the end of this game, $\mathcal{A}$ determines whether it is the actual session key SK or a random value by means of a $Test(Z)$ query. $\mathcal{A}$ can only intercept messages M_1, M_2, M_3, M_4, on the public channel and cannot obtain sensitive values like n_1,n_2,n_3,SID_j. It cannot compute SK based on the acquired messages. therefore, the eavesdropping attack cannot increase the probability of winning GM_1 of the probability. We can obtain:

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

Game GM_2. Adding $Send(Z, I)$ query to game GM_1, according to Zipf's law [32]:

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

Game GM_3. Compared with GM_2, GM_3 adds $Hash(string)$ query and removes $Send(Z, I)$ query. We can get:

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

Game GM_4. In game GM_4, two events are analyzed for security. One is to get x_{GW} of CS, and the second is to get the impermanent information, which proves that our protocol can prevent the temporary information leakage attack.

1. Perfect forward secrecy. $\mathcal{A}$ obtains the long-term key x_{GW} of CS with $\prod_{CS}^{z}$ or $\prod_{U_i}^{x}$, $\prod_{S_j}^{y}$ to obtain the secret parameters of the registration phase.
2. Temporary information leakage attack: $\mathcal{A}$ use $\prod_{CS}^{z}$ or $\prod_{U_i}^{x}$, $\prod_{S_j}^{y}$ to obtain random numbers of three entities.

For the first one, even if $\mathcal{A}$ can obtain x_{GW} of CS, or the secret parameter, the values of n_1,n_2,n_3,SID_j cannot be computed, and the SK cannot be computed. For the second one, even though $\mathcal{A}$ obtains n_1, the values of n_2, n_3, and SID_j are not known and SK cannot be computed. Therefore, we can obtain:

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

Game GM_5. In game GM_5, $\mathcal{A}$ uses $Corrupt(Z)$ query to obtain information in memory $\{A_1, \tau_i, TID_i, V_1\}$ to prove that the protocol prevent information leakage in memory. When the user registers $\mathcal{A}$ wants to surmise $V_1 = h(ID_i \parallel PW_i \parallel \sigma_i \parallel TID_i)$, but the probability of $\mathcal{A}$ guessing information is $1/2^l$. So, the probability of being able to guess the password when $q_{send} \leq 106$, $\mathcal{A}$ is greater than 0.5, so we can obtain:

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

Game GM_6. To verify that the protocol satisfies the simulation attack, $\mathcal{A}$ queries $h(n_1 \oplus n_2 \oplus n_3 \oplus SID_j)$ and the game ends. It is possible to obtain:

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

Since the likelihood of success and failure of a is identical, $\mathcal{A}$ can estimate that the likelihood of the session key is:

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

According to the above formula, we can get:

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

So, we can get:

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

The above proof process implies that $\mathcal{A}$ has no additional advantage to win the game and therefore the protocol is safe under the ROR model.

```
(* ----- Ui's process ----- *)
let ProcessUi =
    new IDi:bitstring;
    new PWi:bitstring;
    let HPWi=h(con(IDi,PWi)) in
    out(sch,(IDi,HPWi));
    in(sch,(xTIDi:bitstring,xA1:bitstring));
    new BIOi:bitstring;
    let (a: bitstring, b: bitstring)=Gen(BIOi) in
    let V1=h(con(con(con(IDi,PWi),a),xTIDi)) in
    !
    (
    event UserStarted();
    let a=Rep(BIOi,b) in
    let V1'=h(con(con(con(IDi,PWi),a),xTIDi)) in
    if V1'=V1 then
    new n1:bitstring;
    new T1:bitstring;
    new SIDj:bitstring;
    let HPWi=h(con(IDi,PWi)) in
    let ai=xor(xA1,HPWi) in
    let A2=xor(n1,h(con(ai,xTIDi))) in
    let A3=xor(SIDj,h(con(HPWi,T1))) in
    let V2=h(con(con(con(ai,n1),SIDj),T1)) in
    out(ch,(A2,A3,V2,T1,xTIDi)); (* ----- authentication ----- *)
    event UserAuthed();
    in(ch,(xTIDinew':bitstring,xA6:bitstring,xV5:bitstring));
    let TIDinew=xor(xTIDinew',h(con(n1,ai))) in
    let p=xor(xA6,h(con(n1,SIDj))) in
    let SKi=xor(xor(n1,p),SIDj) in
    let V5'=h(con(con(SKi,p),TIDinew)) in
    if V5'=xV5 then event UserAcControlSever();

    0 (* ----- authentication ----- *)
    ).
```

FIGURE 5. User's process

5.1.2. *Security authentication with ProVerif.* ProVerif [33, 34] is an automatic cryptographic protocol verification program that can handle a spacious variety of cryptographic primitives, such as hash functions, Diffie-Hellman cryptographic protocols, symmetric and asymmetric cryptographic algorithms, etc. It has been widely used for the formal analysis of cryptographic protocols. In this section, ProVerif is used to verify the safety of our protocol mutual authentication and session keys.

1. We define two channels *sch* and *ch*, *sch* represents the secure channel, and *ch* represents the public channel, which is used for information transfer. Some operations and queries can be clearly seen in Figure 8(a), (b), (c).
2. Figures 5-7 shows the detailed process of participating entities.
3. The whole protocol's representation and the copy results are shown in Figure 8(d). Therefore, the information transmission is safe and the three parties can authenticate each other.

5.2. **Informal security analysis.** In this section, we show that DACSC satisfies the following security.

1. Internal attack: When $\mathcal{A}$ is internal to the server, it is able to get the user's data $\{TID_i, A_1, PSID_j, b_j\}$ which stored in the server database. However, a_i, K_{SG} are

```
(*----Sj proces----*)
let ProcessSj=
in(sch,(yPSIDj:bitstring,ySIDj:bitstring,yKSG:bitstring));
!(
in(ch,(yA2:bitstring,yA3:bitstring,yV2:bitstring,yT1:bitstring,yTIDi:bitstring));
new n2:bitstring;
new T2:bitstring;
let A4=xor(n2,h(con(ySIDj,yKSG))) in
let V3=h(con(con(n2,ySIDj),T2)) in
out(ch,(yA2,yA3,yV2,yT1,yTIDi,A4,V3,T2,yPSIDj));
in (ch,(yPSIDjnew':bitstring,yTIDinew':bitstring,yA5:bitstring,yA6:bitstring,yV4:bitstring,yV5:bitstring));
let PSIDjnew=xor(yPSIDjnew',h(con(ySIDj,n2))) in
let q=xor(yA5,h(con(con(n2,ySIDj),yKSG))) in
let SKj=h(con(con(n2,q),ySIDj)) in
let V4'=h(con(con(SKj,q),PSIDjnew)) in
if V4'=yV4
then event CloudServerAcControlServer();
0).
```

FIGURE 6. Cloud server's process

```
(*---CS's process---*)
let UiReg =
in(sch,(zIDi:bitstring,zHPWi:bitstring));
new ai:bitstring;
let TIDi=h(con(zIDi,ai)) in
let A1=xor(ai,zHPWi) in
out(sch,(TIDi,ai));
0.(* -----Wi registration ----- *)

let SjReg =
new SIDj:bitstring;
new bj:bitstring;
new xGW:bitstring;
let PSIDj=h(con(SIDj,bj)) in
let KSG=h(con(SIDj,xGW)) in
out(sch,(PSIDj,SIDj,KSG));
0.(* -----Sj registration ----- *)

let CSAuth =
in(ch,(zA2:bitstring,zA3:bitstring,zV2:bitstring,zT1:bitstring,zTIDi:bitstring,zA4:bitstring,zV3:bitstring,zT2:bitstring,zPSIDj:bitstring));
new A1:bitstring;
new ai:bitstring;
let HPWi=xor(ai,A1) in
let n1=xor(zA2,h(con(ai,zTIDi))) in
let SIDj=xor(zA3,h(con(HPWi,zT1))) in
(*-------CS  verifies Ui----------*)
let V2'=h(con(con(con(ai,n1),SIDj),zT1)) in
if V2'=zV2 then event ControlServerAcUser();
(*-------CS verifies Sj----------*)
new KSG:bitstring;
let n2=xor(zA4,h(con(SIDj,KSG))) in
let V3'=h(con(con(n2,SIDj),zT2)) in
if V3'=zV3 then  event ControlServerAcCloudServer();
new n3:bitstring;
let SKs=h(con(con(con(n1,n2),n3),SIDj)) in
let PSIDjnew=h(con(zPSIDj,n1)) in
let PSIDjnew'=xor(PSIDjnew,h(con(SIDj,n1))) in
let TIDinew=h(con(con(zTIDi,n1),ai)) in
let TIDinew'=xor(TIDinew,h(con(n1,ai))) in
let q=xor(n1,n3) in
let p=xor(n2,n3) in
let A5=xor(q,h(con(con(n2,SIDj),KSG))) in
let A6=xor(p,h(con(n1,SIDj))) in
let V4=h(con(con(SKs,q),PSIDjnew)) in
let V5=h(con(con(SKs,p),TIDinew)) in
out(ch,(PSIDjnew',TIDinew',A5,A6,V4,V5));
0.
```

FIGURE 7. Control server's process

not stored in the database, $\mathcal{A}$ cannot calculate n_1 and n_2 even if it gets the data in the database. Therefore, our presented protocol can oppose the internal attack.

2. Cloud server node impersonation attack: Suppose $\mathcal{A}$ obtains the messages $M_3 = \left\{PSID_j^{new'}, TID_i^{new'}, A_5, A_6, V_4, V_5\right\}$ transmitted by the cloud server node on the common channel. If $\mathcal{A}$ wants to impersonate a legitimate cloud server node, $\mathcal{A}$ needs to construct the legitimate parameter V_2, but $\mathcal{A}$ cannot get a_i by the known attacker model and cannot compute V_2. Therefore, this protocol can oppose cloud server node imitation attacks.
3. User impersonation attack: Suppose $\mathcal{A}$ obtains the messages $\{A_2, A_3, V_2, T_1, TID_i\}$, if $\mathcal{A}$ wants to impersonate a legitimate user, $\mathcal{A}$ needs to construct the legitimate

```
(* 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 SKs:bitstring [private].

(* constants *)
free s: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.

fun Gen(bitstring):bitstring. (*Generator operation *)
fun Rep(bitstring,bitstring):bitstring.
```

(a)Definition

```
(* queries *)
query attacker(SKi).
query attacker(SKj).
query attacker(SKs).
query inj-event(UserAuthed())==>inj-event(UserStarted()).
query inj-event(ControlServerAcUser())==>inj-event(ControlServerAcCloudServer()).
query inj-event(CloudServerAcControlServer())==> inj-event(UserAcControlSever()).

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

(b)Events

```
let ProcessCS  = UiReg | SjReg | CSAuth.

(* ----- Main ----- *)
process
    (!ProcessUi | ! ProcessSj| !ProcessCS)
```

(c)Main

```
--------------------------------------------------------------
Verification summary:

Query not attacker(SKi[]) is true.

Query not attacker(SKj[]) is true.

Query not attacker(SKs[]) is true.

Query inj-event(UserAuthed) ==> inj-event(UserStarted) is true.

Query inj-event(ControlServerAcUser) ==> inj-event(ControlServerAcCloudServer) is true.

Query inj-event(CloudServerAcControlServer) ==> inj-event(UserAcControlSever) is true.

--------------------------------------------------------------
```

(d)Results

FIGURE 8. Definition and Results

parameter V_2, but Adv cannot get a_i through the known attacker model to calculate V_2. Therefore, this protocol can resist this attack.

4. Replay attack: In this protocol, random values or time stamps are added when transmitting information, and the attacker cannot obtain n and T while the session is in progress, so even if the attacker can get the messages transmitted on the public channel, he cannot perform a replay attack, therefore, this protocol is able to resist replay attacks.
5. Mutual authentication: Using the values of S_j and U_i, V_4 and V_5 are utilized to mutually authenticate the identity while CS verifies participating entities using V_2 and V_3, respectively. Although V_2 and V_3 are transmitted on a common channel, the adversary cannot obtain the value of a_i. Similarly, V_4 and V_5 are transmitted on a common channel, but the adversary cannot obtain the values of n_1, n_2, n_3 and SK, so the protocol cannot be broken by changing the authentication values. The code of behavior proposed in this paper can do mutual authentication.
6. User anonymity: The attacker $\mathcal{A}$ tries to identify the user by intercepting the message on the public channel. In the login and authentication phases, the user's ID_i is not directly transmitted over the public channel, but the user's pseudo-identity TID_i is transmitted. thus the adversary only obtains TID_i and does not get the user's real identity ID_i. Our proposed protocol enables user anonymity.

Table 2. Comparisons of security

Security Properties	[11]	[13]	[20]	[10]	DACSC
User Anonymity	✓	×	✓	✓	✓
Insider Attack	×	✓	✓	✓	✓
Mutual Authentication	✓	×	✓	✓	✓
Impersonation Attack	×	×	✓	✓	✓
Replay Attack	✓	×	✓	✓	✓
Session-specific Ad Hoc Information Attack	✓	✓	×	✓	✓
Offline Password Guessing Attack	✓	×	×	✓	✓
Perfect Forward Security	✓	✓	✓	✓	✓
Dynamic Authentication Credentials	×	×	×	×	✓

6. **Safety and performance comparison.** In this section, we contrast with other associated schemes [10,11,13,20] in terms of computational overhead, communication overhead and security in the login and authentication phases. The specific comparisons are described below.

6.1. **Safety comparison.** Our protocol has better security compared to existing protocols, and Table 2 shows the results of the analysis of security features, where × indicates that this security feature is not satisfied and ✓ indicates that this security feature is satisfied.

Table 2 shows that the protocol of Amin et al. [11] cannot resist internal attacks, spoofing attacks, the protocols of Pelaez et al. [13] and Kang et al. [20], cannot resist various attacks such as spoofing attacks, replay attacks, and mutual authentication, and the protocol of Yu et al. [10] cannot dynamically update authentication credentials, our proposed protocol is clearly resistant to various attacks.

6.2. **Performance cost.**

6.2.1. *Computational cost.* We compare the computational overhead of this scheme with the existing schemes [10, 11, 13, 20], and the comparison results are shown in Table 3 , Table 4 and Fig 9.

Table 3 shows that the user computational costs of the Amin et al. [11] scheme takes 0.0092ms, the Pelaez et al. [13] scheme takes 0.2112ms, the Kang et al. [20] scheme takes 0.0083ms, the Yu et al. [10] scheme 0.0122ms, and our scheme takes 0.0148ms, the user computation price is a little higher than Amin et al. [11], Kang et al. [20], Yu et al. [10] is slightly higher, but it is better than them in terms of security.

It can be seen from Table 5 that the computational cost of our protocol server is smaller than that of Pelaez et al. [13] and Yu et al. [10], and only 0.0312ms higher compared to Amin et al. [11] and Kang et al. [20], giving a higher security at a lower cost.

T_h, T_b, T_{de}, T_{en} denote the computation time of hash function, fuzzy function, symmetric decryption, symmetric encryption, respectively, and XOR computation can be neglected. Referring to the work of [35], the computation time T_h, T_b, T_{de}, T_{en} for users on an Android phone with Qualcomm Xiaolong 865 CPU and 8G running memory are 0.00102ms, 0.00561ms, 0.2ms, and 0.0591ms, respectively. On a computer with windows 10 operating system, Intel(R) Core(TM) i5-8500CPU@3.00GHz3.00G CPU processor and 8GB RAM, the server's computation time T_h, T_{de}, T_{en} are 0.0052ms, 0.1347ms, and 4.7ms, respectively.

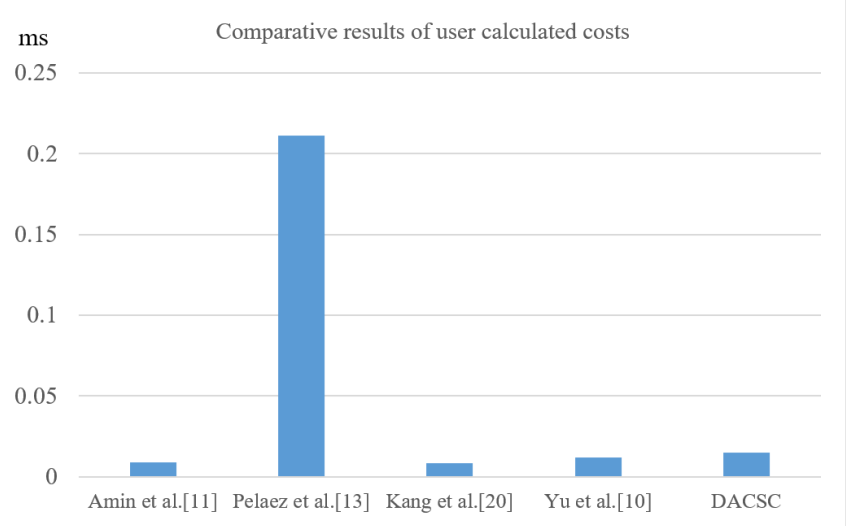

FIGURE 9. Comparative results of user computational costs

TABLE 3. User Computational Costs

Agreement	User	Total(ms)
Agreement	$9T_h$	0.0092
Pelaez et al.	$11T_h + T_{de}$	0.2112
Kang et al.	$8T_h$	0.0083
Yu et al.	$12T_h$	0.0122
DACSC	$9T_h + T_b$	0.0148

TABLE 4. Server Computational Costs

Agreement	Cloud Server	Control Server	Total(ms)
Amin et al	$4T_h$	$10T_h$	0.0728
Pelaez et al.	$6T_h + 2T_{de} + T_{en}$	$34T_h + 2T_{en}$	14.5774
Kang et al.	$3T_h$	$11T_h$	0.0728
Yu et al.	$6T_h$	$16T_h$	0.1144
DACSC	$6T_h$	$14T_h$	0.104

TABLE 5. Computational rounds and Costs

Agreement	Communication rounds	Communication costs (bit)
Amin et al	5	3392
Pelaez et al.	6	4448
Kang et al.	2	3712
Yu et al.	4	3200
DACSC	5	3872

6.2.2. *Communication cost.* To calculate the communication cost, we assume that the one-way hash function H, a random number n, a string s, and an identity ID are 256 bits, 160 bits, 160 bits, 160 bits, respectively, the timestamp T is 32 bits, and the encryption operation E is set to 256 bits.

Amin et al. [11] have a storage cost of $3|ID|+8|s|+6|H|+3|T|$, while Pelaez et al. [13] have $3|ID|+18|s|+|H|+2|E|+2|n|$. Kang et al. [20] have $3|ID|+10|s|+6|H|+3|T|$,

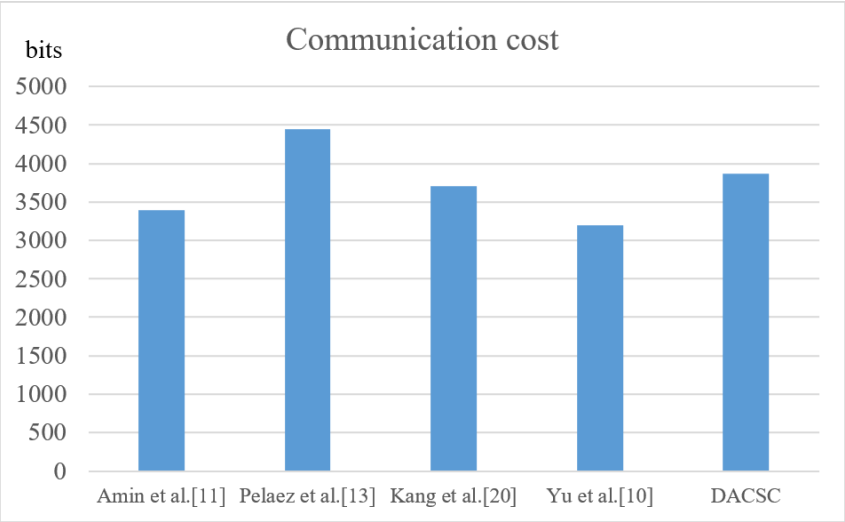

FIGURE 10. Comparative results of communication costs

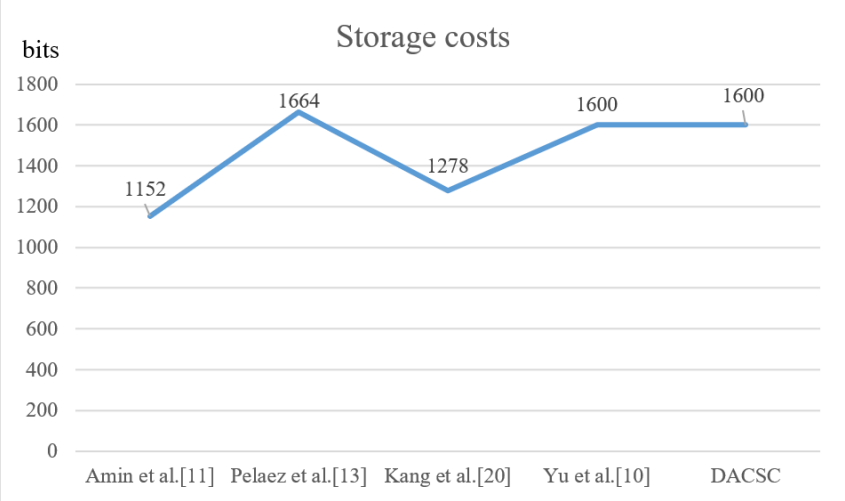

FIGURE 11. Comparative results of communication costs

and Yu et al. [10] have $3|ID| + 9|s| + 5|H|$. However, the storage cost of our DACSC is $3|ID|+11|s|+6|H|+3|T|$. The detailed comparison results are shown in Table 3 and Fig 10. In comparison, the communication cost of our protocol is much smaller than that of Pelaez et al. [13], although it is slightly higher than the communication cost of Amin et al. [11] Kang et al. [20] and Yu et al. [10], but their protocols have various security issues that are not applicable for key negotiation in cloud computing environment. Our protocol consumes only a small amount of cost in exchange for better security. We calculated the registration cost for storing all parameters in the registration phase, as shown in Fig 11. It can be seen that our protocol does not consume too much storage cost.

7. **Conclusions.** Cloud computing is a new service model that enables every user using the Internet to store huge amounts of data, but a large amount of private data is transmitted through public channels and stored in cloud servers, which will cause irreversible consequences such as economic losses to individuals and enterprises if leaked. In this paper, we design an authentication and key negotiation scheme based on DAC and IntelSGX in cloud computing environment. Our scheme implements dynamic update of authentication credentials to ensure successful negotiation of session keys, and uses SGX

to store the master key as well as some identity information to resist man-in-the-center attacks, simulation attacks, etc.

In addition, we made a detailed comparison with other existing studies in terms of safety and cost. The comparison results show that our protocol makes better progress than other schemes. Although the cost of our protocol is not the least, the security of other schemes is vulnerable to certain specific attacks and cannot meet the needs. It is necessary to exchange a small amount of cost for higher security in the era of rapid development of information technology. We use ProVerify model to prove that our protocol is secure. Therefore, our scheme has better security and is more efficient and appropriate for cloud computing environment.

8. **Acknowledgments.** This work was supported in part by the National Science Foundation of Shandong Province under the Grant ZR2022MF338 and ZR2022LZH014, Humanity and Social Science Fund of the Ministry of Education under Grant 20YJAZH078 and 20YJAZH127, Open Project of Tongji University Embedded System and Service Computing of Ministry of Education of China under Grant ESSCKF2022-02.

REFERENCES

[1] M. Shen, X. Tang, L. Zhu, X. Du and G. M, Privacy-preserving support vector machine training over blockchain-based encrypted IoT data in smart cities, *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7702–7712, 2019.

[2] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami and R. R. Zebari, IoT and Cloud computing issues, challenges and opportunities: A review, *Qubahan Academic Journal*, vol. 1, no. 5, pp. 1–7, 2021.

[3] L. N. Ni, X. T. Sun, X. C. Li and J. Q. Zhang, Computational Intelligence and Neuroscience, *Computational Intelligence and Neuroscienced*, vol. 2021, pp. 1-17, 2021.

[4] L. N. Ni, P. Huang, Y. S. Wei, M. L. Shu and J. Q. Zhang, Federated learning model with adaptive differential privacy protection in medical IoT, *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1-14, 2021.

[5] J. Q. Zhang, H. R. Liu and L. N. Ni, A secure energy-saving communication and encrypted storage model based on RC4 for EHR, *IEEE Access*, vol. 8, pp. 38995–39012, 2020.

[6] L. N. Ni, J. Q. Zhang, C. J. Jiang, C. G. Yan and K. Yu, Resource allocation strategy in fog computing based on priced timed petri nets, *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[7] J. K. Zhang, T. T. Wu, X. T. Sun and J. G. Yu, MPDP k-medoids: Multiple partition differential privacy preserving k-medoids clustering for data publishing in the Internet of Medical Things, *International Journal of Distributed Sensor Networks*, vol. 17, no. 10, pp. 15501477211042543, 2021.

[8] C. Y. Wang , K. Ding, B. Li, Y. M. Zhao, G. A. Xu, Y. H. Guo and P. Wang, N-in-One: A Novel Location-Based Service, *Wireless Communications*, vol. 2018, 2018.

[9] X. Li, J. W. Niu, S. Kumari , F. Wu, A. K. Sangaiah and K. R. Choo, A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments, *Journal of Network and Computer Applications*, vol. 103, pp. 194-204, 2018.

[10] S. J. Yu, K. S. Park, Y. H. Park, A secure lightweight three-factor authentication scheme for IoT in cloud computing environment, *Sensors*, vol. 19, no. 16, pp. 3598, 2019.

[11] R. Amin, N. Kumar, G. P. Biswas, V. Chang and R. Iqbal, A light weight authentication protocol for IoT-enabled devices in distributed Cloud Computing environment, *Future Generation Computer Systems*, vol. 78, pp. 1005-1019, 2018.

[12] L. Zhou, X. Li, K. Yeh, C. Su and W. Chiu, Lightweight IoT-based authentication scheme in cloud computing circumstance, *Future generation computer systemss*, vol. 2018, pp. 244-251, 2019.

[13] R. Martínez-Peláez, H. Toral-Cruz, J. R. Parra-Michel, V. García, L. J. Mena, V. G. Félix and A. Ochoa-Brust, Achieving differential privacy of trajectory data publishing in participatory sensing, *Sensors*, vol. 19, no. 9, pp. 2098, 2019.

[14] X. Liu, R. S. Zhang, M. Q. Zhao, A robust authentication scheme with dynamic password for wireless body area networks, *Computer Networks*, vol. 1761, pp. 220-234, 2019.

[15] V. Costan, S. Devadas, Intel SGX explained, *Cryptology ePrint Archive*, 2016.

[16] K. Xue, P. Hong, C. Ma, A lightweight dynamic pseudonym identity based authentication and key agreement protocol without verification tables for multi-server architecture, *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 195-206, 2014.

[17] S. Challa, A. K. Das, P. Gope, N. Kumar, F. Wu, A. V. Vasilakos, Design and analysis of authenticated key agreement scheme in cloud-assisted cyber–physical systems, *Future Generation Computer Systems*, vol. 108, pp. 1267-1286, 2020.

[18] J. L. Tsai, N. W. Lo, A privacy-aware authentication scheme for distributed mobile cloud computing services, *IEEE Systems Journal*, vol. 9, no. 3, pp. 805-815, 2015.

[19] D. He, N. Kumar, M. K. Khan, L. Wang, J. Shen, Efficient privacy-aware authentication scheme for mobile cloud computing services, *IEEE Systems Journal*, vol. 12, no. 2, pp. 1621-1631, 2016.

[20] B. Y. Kang, Y. H. Han, K. Qian, J. Q. Du, B. Li, Analysis and improvement on an authentication protocol for IoT-enabled devices in distributed cloud computing environment, *Mathematical Problems in Engineering*, vol. 2020, 2020.

[21] Z. Li, L. Yang, T.-Y. Wu, C. M. Chen, Cryptanalysis of an Authentication Protocol for IoT-Enabled Devices in Distributed Cloud Computing Environment, *Advances in Intelligent Systems and Computing: Proceedings of the 7th Euro-China Conference on Intelligent Data Analysis and Applications, May 29–31, 2021, Hangzhou, China*, pp. 339-347, 2022.

[22] R. A. Balisane, A. Martin, Trusted execution environment-based authentication gauge (TEEBAG), *Proc. IEEE 2013 10th Inter. Conf. on Ubiquitous Intelligence and Computing and Autonomic and Trusted Computing* , pp. 61-67, 2016.

[23] R. C. Condé, C. A. Maziero, N. C. Will, Using Intel SGX to protect authentication credentials in an untrusted operating system, *2018 IEEE Symposium on Computers and Communications (ISCC)*, pp. 200158-00163, 2018.

[24] J. Wang, Y. Jiang, Q. Li, Y. Yang, Survey of research on SGX technology application, *Journal of Network New Media*, vol. 6, no. 5, pp. 3-9, 2017.

[25] P. Jain, S. Y. Desai, M. Shih, T. Kim, S. M. Kim, J. Lee, C. Choi, Y. J. Shin, B. B. Kang, D. Han, OpenSGX: An Open Platform for SGX Research, *NDSS*, vol. 16, pp. 21-24, 2016.

[26] Y. Ding, R. Duan, L. Li, Y. Q. Cheng, Y. L. Zhang, T. H. Chen, T. Wei, H. Wang, POSTER: Rust SGX SDK: Towards memory safety in Intel SGX enclave, *Knowledge and Information Systems*, pp. 2491-2493, 2017.

[27] J. Wang, Y. Shi, G. Peng, H. Zhang, B. Zhao, F. Yan, F. Yu, and L. Zhang, Survey on key technology development and application in trusted computing, *China Communications*, vol. 13, no.11, pp. 70-90, 2016.

[28] R. Canetti and H. Krawczyk, Analysis of key-exchange protocols and their use for building secure channels, *Advances in Cryptology?EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6-10, 2001 Proceedings 20s*, pp. 453-474, 2001.

[29] D. Dolev and A. Yao, On the security of public key protocols, *IEEE Transactions on Information Theory*, vol. 29, no.2, pp. 198-208, 1983.

[30] R. Canetti, O. Goldreich and S. Halevi, The random oracle methodology, revisite, *Journal of the ACM (JACM)*, vol. 51, no.4, pp. 557-594, 2004.

[31] T.-Y. Wu, T. Wang, Y. Q. Lee, W. Zheng, S. Kumari and S. Kumar, Improved authenticated key agreement scheme for fog-driven IoT healthcare system, *Security and Communication Networks*, vol. 2021, pp. 1-16, 2021.

[32] D. Wang, H. Cheng, P. Wang, X. Huang and G. Jian, Zipf?s law in passwords, *IEEE Transactions on Information Forensics and Security*, vol. 12, no, 11, pp. 2776-2791, 2017.

[33] B. Blanchet, A computationally sound mechanized prover for security protocols, *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 4, pp. 193-207, 2008.

[34] M. Abadi and C. Fournet, Mobile values, new names, and secure communication, *ACM Sigplan Notices*, vol. 36, no. 3, pp. 104-115, 2001.

[35] T.-Y. Wu, Q. Meng, S. Kumari and P. Zhang, Rotating behind Security: A Lightweight Authentication Protocol Based on IoT-Enabled Cloud Computing Environments, *Sensors*, vol. 22, no. 10, pp. 3858, 2022.