



# Enhancing security and efficiency in cloud computing authentication and key agreement scheme based on smart card

Mariem Bouchaala<sup>1</sup> · Cherif Ghazel<sup>1</sup> · Leila Azouz Saidane<sup>1</sup>

Accepted: 30 April 2021 / Published online: 28 May 2021

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

## Abstract

The password-based authentication mechanism is considered as the oldest and the most used method. It is easy to implement, and it does not require any particular configuration or devices. Yet, this solution does not ensure a high level of security when it is used in a large and remote environment such as cloud computing. In such an environment, the cloud user and the authentication remote server use an insecure communication channel to authenticate each other. Consequently, various attacks such as insider attack, password-guessing attack, user impersonation attack, and others can be launched. Smart cards are an alternative to improve this single authentication model by strengthening security and improving the communication process. In our work, we study the Huang et al. proposal. The authors have proposed a smart card-based authentication and key agreement scheme. They have used the elliptic curve to improve security. However, some related work shows that this solution does not resist to impersonation attacks and does not ensure perfect anonymity. Consequently, it does not protect users' privacy. Thus, we propose an extension of the Huang et al. scheme in order to enforce security requirements. We implement an anonymous, mutual, and secure two-factor authentication and key agreement scheme applied to the cloud computing environment. We use elliptic curve cryptography and a fuzzy verifier to strengthen security. The solution is lightweight and optimizes performance. To prove the safety of the proposed protocol, formal security analysis with random oracle model and Scyther tool is provided. To evaluate its efficiency, a performance evaluation is prepared.

**Keywords** Cloud computing · Smart cards · Authentication · Key agreement · Elliptic curve cryptography · Security · Scyther tool

---

✉ Mariem Bouchaala  
bouchaala.mariem@gmail.com

<sup>1</sup> CRISTAL Laboratory, ENSI, University of Manouba, Manouba, Tunisia

## 1 Introduction

Cloud computing is the most promising paradigm in recent years. It contributes in the appearance and the evolution of new concepts and technologies. It offers on demand a fragrance of measured services accessible through heterogeneous platform and devices. Cloud users outsource their assets to remote cloud servers. Preserving privacy and fine-grained access are important concerns in security fields. We mean by privacy the fact that an eavesdropper cannot intercept exchanged messages. To do so, cloud provider should ensure anonymity and untraceability for users. Authentication ensures that unauthorized user cannot fraudulently access to cloud services.

Authentication protocols can be regrouped to four main fields namely password, cryptography, biometric, and smart cards. All these approaches deal with some limitations due to the capabilities of public and remote environments. In our work, we will focus only in authentication method based on smart cards. Smart cards are considered as a secure micro-controllers that can support cryptographic and storage operation. These cards are increasingly used in information technology and in everyday life. They assure exchange and authentication without using a password management server. Smart cards offer a strong authentication solution. It is considered two-factor by the way that a user should know authentication credentials and should also have the chips. Despite these advantages, smart cards suffer from low computational capacity. Moreover, public key authentication solutions are not supported. It is necessary to apply a low computational method based mainly on hash functions and symmetric encryption operations. For the opportunities offered by these cards, we propose in this work a new anonymous, lightweight, secure, efficient, and smart-card-based authentication and key agreement protocol that withstand various known attacks.

In this paper, we first analyze Huang et al.'s authentication method, and we find that this scheme is insecure against impersonation attacks. It does not ensure anonymity and consequently privacy. Finally, Huang does not detail the possibility of revoking smart cards in case of lost or theft. Then, we propose a new two-factor scheme as an extension of drawbacks raised in Huang solution. Finally, we prove our proposal in terms of security and performance.

The rest of the paper is structured as follows. In Sect. 2, we discuss some related works in remote cloud computing environment. Section 3 studies some preliminaries. Section 4 details the proposed solution. Section 5 provides a formal and informal security verification and validation. Section 6 evaluates the performance of the proposed solution. Brief summary and concluding remarks are presented in sect. 7.

## 2 Related work

In the literature, many authentication approaches are used such as passwords, tokens, digital signatures, smart cards, biometrics. Single-factor authentication (SFA) is a way of using only one category of these approaches.

As first attempt in research field, authors use identity and password [1–3] in authentication process. When a cloud user want to access, he sends the identity and the password to cloud server. If the request matches with one of the available records in the authentication database, user will be identified and access is authorized. In order to secure the authentication request, a hash function is used to slat the user password. Single-factor mechanisms are not strong, and it presents in the most of case a security hole for launching attacks such as offline/online guessing attacks.

Image-based authentication systems [4–6] deal with some mathematical transforms, codes and selections. Authors combine sometimes fractional Fourier transform (FFT) or wavelet packet transform (WPT) and other times Hamming code technique, graphics and image selection with one-time password (OTP). These works offer an efficient and strong authentication solution. Yet, they provide only user authentication mechanisms and does not deal with key agreement between Cloud user and server. As a result, they does not ensure mutual authentication.

Smart cards present a strong authentication model due to the hardness of revealing keys and secrets because they are protected logically and physically. In the literature, a wide number of authentication methods based on smart cards have been proposed. In the case of cloud computing, a cloud server embeds the authentication parameters into a smart car, and then delivers it to the cloud user. Authors in [7] discuss smart card-based authentication scheme. They identify and classify several attacks launched in proposed solution in research fields.

To resolve the issues related to smart cards, DAS proposed in [8] a two-factor authentication model for wireless sensor network (WSN). For this purpose, he uses both password and smart card. He proves that this solution improves security and communication cost but it presents a start point of discussions and works in this field. Papers [9, 10] identify that DAS's scheme suffers from several security issues namely password-guessing attack, impersonation attack, and insider attack.

In 2018, Preeti in [11] relies on Rabin cryptosystem which is based on prime integer factorization. This solution is proved efficient in the terms of performance evaluation. It improves smart card storage cost, communication and computation cost and execution time. Yet, in term of security, the Rabin cryptosystem-based solutions are not yet proved indistinguishable against chosen plaintext attacks.

In [12], authors work on preserving user anonymity in a lightweight two-factor authentication protocol. This solution ensures that the proposed scheme provides the session key (SK) security and resolves the problem of offline password-guessing attacks. In 2018, Trupil [13] reviews Nikooghadam et al.'s scheme and proves that it suffers from replay, password-guessing, and insider attack. Trupil improves the authentication system by the remedy of these attacks.

Elliptic curve cryptography (ECC) provides an efficient solution for the improvement of authentication models. It provides the same security level compared to existing solutions under the constraint of reducing key length and computational cost. For example, 160-bit ECC offers the same security level compared to 1024-bit RSA. Consequently, it is highly deployed in cloud computing environment. For these reasons, we will review in this section different works proposing two-factor authentication solutions based on ECC.

Juan et al. proposed in [14] a password and smart card authentication solution with key agreement using elliptic curve. This proposal overcomes many security weaknesses. However, several works provide a cryptanalysis of this solution and prove that it does not withstand some security attacks. In 2015, Huang et al. [15] showed that Juan's solution suffers from impersonation, smart card loss, and password-guessing attacks. They propose a new enhanced key agreement solution for authentication. In 2016, Chaudhry et al. [16] demonstrated that this scheme has a correctness issues. Furthermore, they suggested an improved solution to overcome the security issues. In addition, Maitra et al. proposed an extension of Huang proposal to solve some issues [17]. They present an ECC-based mutual authentication scheme. The proposed solution ensures efficient login and authentication phases and makes possible password update. Unfortunately, Wang et al. showed that solution proposed in [17] does not resist offline password-guessing attack and insider attack, and it does not ensure perfect forward secrecy. To overcome those issues, Wang et al. [18] proposed an improved construction integrating the discrete logarithm problem with RSA cryptosystem to achieve security goals. In 2019, authors in [19] analyzed Wang solution [18] and showed that two major security attributes are not verified namely resisting to offline password-guessing attack and impersonation attack. Consequently, an extension based on elliptic curve cryptography (ECC) was proposed. Authors assume that all exchanged messages in registration, authentication, and login phases are sent over a public communication channel. Yet, the proposed solution greatly increases communication and computation cost.

Independently, Chou et al. [20] dealt with the problem of secure communication between interactive parties. Authors apply elliptic curve cryptosystems in a two-identity key agreement and exchange solution. In 2014, Farash et al. [21] showed that Chou's solution is vulnerable to impersonation and key-compromise attacks. They propose an improved identity-based scheme based on ECC and resolve these issues. Lu et al. [22] enhanced the security issues presented in Farash's solution. They proposed an improved anonymous scheme based on ECC to eliminate key-compromise masquerading and off-line guessing attacks.

However, these previous solutions does not ensure privacy and user anonymity. Memon et al. [23] ensured user location privacy by using asymmetric cryptography scheme. They propose an anonymous communication for location

based service. In 2016, work [24] proved that Memon's solution is insecure against compromised impersonation and insider attack. It provides an imperfect mutual authentication. Consequently, Reddy et al. [24] implemented an enhanced secure and two-factor authentication protocol. This solution provides a perfect mutual authentication with key-agreement phase. Xi et al. [25] proposed the use of a dynamic user ID in two-factor authenticated solution. They introduce in addition revocation and password update algorithms. Authors claimed that their solution ensures user anonymity and overcomes smart card loss attack.

In recent years, the use of smart card-based systems for mutual authentication and key agreement solution devolves considerably. This approach is applied in various fields in particular telecare medical information systems [26], Internet of Thing (IoT) [27, 28], etc. For this reason, hardware producers work to improve devices and make keyboards with built-in card readers.

In summary, the use of smart cards, Elliptic-curve and others cryptography operations in authentication protocols for remote computing environment presents a promising axis in research fields. It improves significantly security and provides key agreement approaches. Yet, they require a third party between cloud user and the cloud server. For this reason, the communication and computation overhead of the overall system is substantially higher. We add also that such system does not preserve privacy for users. It tracks some sensitive information such as identity and location that maybe deployed by attackers. In order to resolve the aforementioned issues, we will implement an efficient authentication and key agreement solution for cloud computing.

### 3 Elliptic curve cryptography

An elliptic curve  $E : y^2 = x^3 + ax + b \pmod p$  represents the set of limited keys  $E_p(a, b)$  that adjust the discriminant  $\Delta = 4a^3 + 27b^2 \pmod p$ . Given  $P$  a specified point in Keys solution and  $n$  a multiplier, the scalar multiplication is computed as  $P \cdot n = P + P + P + \dots + P$  ( $n$  times).

The strength of ECC security comes back to these following mathematical problems:

1. Given  $P$  and  $Q$  two arbitrary points  $\in E_p(a, b)$ , the elliptic curve computational Diffie–Hellman logarithm problem (ECDLP) is to find an integer  $n \in Z_p^*$  such that  $P = nQ$ .
2. Given three points  $P, n.P$  and  $m.P \in E_p(a, b)$  for  $n$  and  $m \in Z_p^*$ , the computational Diffie–Hellman problem (CDLP) is to find the point  $(mn)P \in E_p(a, b)$ .
3. Given two points  $P$  and  $Q$ , with  $Q = n.P + m.P \in E_p(a, b)$  the elliptic curve factorization problem (ECFP) is to find two points  $n.P$  and  $m.P$ .

For the purpose of security analyses of Huang protocol, we expose valid assumptions used in authentication model. We will use the common adversial model used in [29, 30].

**Definition 1** The adversary  $A$  can control all messages transmitted through public communication link.  $A$  can intercept, interrupt, delete, modify, and retransmit a new forged message.

**Definition 2** Adversary  $A$  can retrieve the smart card information by conducting power consumption monitoring methods.

**Definition 3** The registered user always uses dictionary word to define password and identity. *The adversary*  $A$  cannot retrieve user identity and password within polynomial time even if he can intercept  $h(ID||PW)$ . Based on work in [30], the probability of guessing user password is  $\frac{1}{2^{6n}}$  with  $n$  presents the number of password characters.

**Definition 4** In cryptography, we assume that secret key and random are large enough. Consequently,  $A$  cannot guess these information in polynomial time.

**Definition 5** Let  $B = B_1 \oplus B_2$ .  $A$  cannot find  $B_1$  and  $B_2$  in polynomial time even if he can retrieve the value of  $B$ .

After analyzing Huang et al.'s authentication protocol and based on solution proposed in [17], we deduce that the scheme is exposed against impersonation attack and suffer from incorrect notion of perfect anonymity. Let  $A$  be a malicious user. After registration,  $A$  performs following steps:

- Step 1* Let  $A$  a smart card with  $AID_A$ ;  $BID_A$  and  $r_A$ . By using  $PW_A, ID_A$  and the stored values in the smart card, the adversary computes  $TID_A = AID_A - H_1(PW_A || ID_A || r_A) \cdot O$  and  $XID_A = TID_A \cdot H_1(PW_A || ID_A || r_A)^{-1} = H_1(msk) \times O$ .
- Step 2* The adversary firstly selects a random  $y$  and computes  $Y = y \times O$ . Then,  $A$  he computes  $M_A = y \cdot mpk$ ,  $CID_A = H_4(ID_{cu} || M_A) \oplus H_2(M_A || TID_A)$ ,  $EID_A = H_3(H_4(ID_{cu} || M_A) || Y || M_A)$ . Next, he selects an arbitrary number  $alea$  with the same length of hash function  $H_1$  and computes  $DID_A = M_A \oplus alea \times O$ . Finally, he sends  $\langle CID_A, DID_A, EID_A, Y \rangle$  to cloud server.
- Step 3* The server will compute in turn these following equations:  $M'_A = Y \cdot mpk$ ,  $alea \cdot O = DID_A \cdot M'_A$ ,  $TID'_A = H_1(msk) \cdot (DID_A \oplus M'_A)$ ,  $H_4(ID_{cu} || M'_A) = CID_A \oplus (M'_A || TID'_A)$  and  $EID'_A = H_3(H_4(ID_{cu} || M'_A) || Y || M'_A)$ .
- Step 4* The cloud server verifies  $EID'_A = EID_A$ . If it holds, it generates a random number  $r_A$  and computes  $R_A = r_A \times Y$ ,  $T = (R_A \oplus M_A)$ ,  $H_A = H_3(EID'_A || R_A || TID'_A)$ . Finally, it sends to the user  $R_A$  and  $H_A$ .
- Step 5* The adversary will compute  $T'_A = R_A \oplus M_A$  and  $H'_A = H_3(EID_A || T'_A || TID_A)$ .

**Conclusion** The adversary  $A$  has impersonated successfully the server. It succeeded in the phase of negotiation and generation of a shared session key based on the identity of the user  $cu$ . This key is computed as follows:  $SK = H_5(R_A || Y || M_A || TID_A)$ . We add also that user's authentication credentials are not protected efficiency. A malicious user can intercept the user identity. Hence, anonymity and privacy are not ensured. Finally, we add also that the aforementioned solution does not deal with the case of smart cards lost or theft. They does not propose a revocation algorithm.

#### 4 Proposed smart card-based authentication scheme description

In this section, we will propose an improvement of the solution proposed in [15]. Our construction is composed by five main algorithms in particular setup, registration, mutual authentication, password update and card revocations. To do so, we will

expose notations used in these algorithms. After that, we expose security requirements. Finally, we develop all algorithms.

### 4.1 Security requirements

In our construction, we define these assumptions.

- We assume that in registration phase cloud server and cloud user are honest.
- In registration phase, all information provided by both parties are correct and trusted.
- In registration phase, the channel assuring communication between user and server is secure.
- In authentication phase, the channel assuring communication between user and server is no longer secure.
- Smartcards and communications standards are secure.
- Both server and user should authenticate each others before authorizing access.
- Cloud sever must not get the user password in any condition.
- In password update phase, the cloud user can change at any time the password seamlessly without informing the cloud server.
- Any password in a correct format should be taken.

### 4.2 Notations

Some notations related to elliptic curve and cryptography of our solution are announced in Table 1.

Table 1 Notations

Notations	Descriptions
$p, q$	large prime number
$F_p$	prime field
$E_q$	q-order elliptic curve
$Z_q^*$	values $\in \{1, 2, 3, \dots, q - 1\}$
$S_k$	Cloud server secret key
PP	Public parameter
$P_g$	Generator of an additive group G
$ID_{card}$	Smart card identity
$ID_u$	User identity
$Pass_u$	User password
$ID_s$	server identity
$h(.) \{0, 1\}^* \{0, 1\}^k$	one-way hash function
$T_i$	registration time stamp
$SK_x$	Session Key

### 4.3 Setup phase

This algorithm generates public and secret parameters used in other algorithms. It outputs master secret key, public key, and fuzzy parameter.

- Step 1* Fix on an elliptic curve  $E_q$  over a finite field  $F_p$ .
- Step 2* Choose a master key  $s_k \in Z_q^*$  and compute the public parameter  $PP = s_k P_g$ .
- Step 3* Publish public parameters  $\langle F_p, E_q, P_g, PP, G, h \rangle$  and kept in secret master key  $s_k$ .
- Step 4* Select a fuzzy parameter **fuzzy**  $\in [2^4, 2^8]$ . This value extends classical Boolean logic with partial truth values. We use this parameter to have a wide set of real instead of 0 and 1. Thereafter, **fuzzy** allows to increase the complexity of computation.

### 4.4 Registration phase

In this section, we expose bidirectional exchange information between a cloud user with identity  $ID_u$  and a remote cloud server. Figure 1 exposes the different corresponding steps.

- Step 1* The user selects a random  $r_i$  and computes  $PW = h(r_i \parallel PASS_u)$  in order to ensure collusion and preimage resistance.
- Step 2* Remote cloud server chooses a random  $b_i$ . It stores the registration time  $T_i$ , the user identity  $ID_u$ , the random  $b_i$  and the smart card number  $ID_{card}$ .
- Step 3* Remote cloud server computes  $A_1 = h(h(ID_u \parallel b_i \parallel T_i \parallel ID_{card}) \text{ mod fuzzy}) \oplus PW$ . It sends a smart cart to cloud user containing  $\langle A_1, ID_s, h() \rangle$ ,  $P_g, PP, \text{fuzzy}$
- Step 4* Cloud user computes and stores into smart card  $A_2 = r_i \oplus h(ID_u) \parallel PW \text{ mod fuzzy}$ .

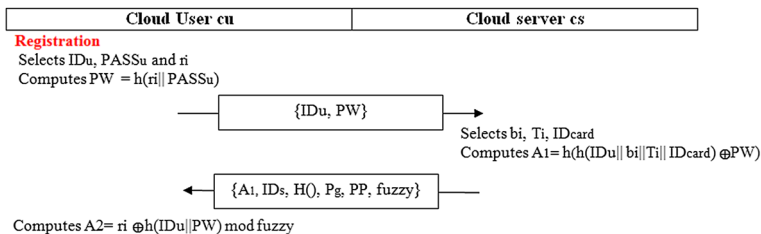


Fig. 1 Proposed registration algorithm



### 4.5 Authentication phase

Figure 2 illustrates the steps of login and authentication of the proposed solution. In our protocol, we divide this party to three sub-algorithms. First, the cloud user uses his smart card and sends an authentication request to the remote cloud server. Then, the cloud server verifies the legitimacy of the received request. Finally, both parties will generate a shared session key after mutual authentication and key negotiation.

#### 4.5.1 Cloud user CU login request

The credentials of a user  $u$  are composed by an identity and a password, respectively, noted by  $ID_u$  and  $PASS_u$ .

*Step 1* The cloud user inserts a smart card.

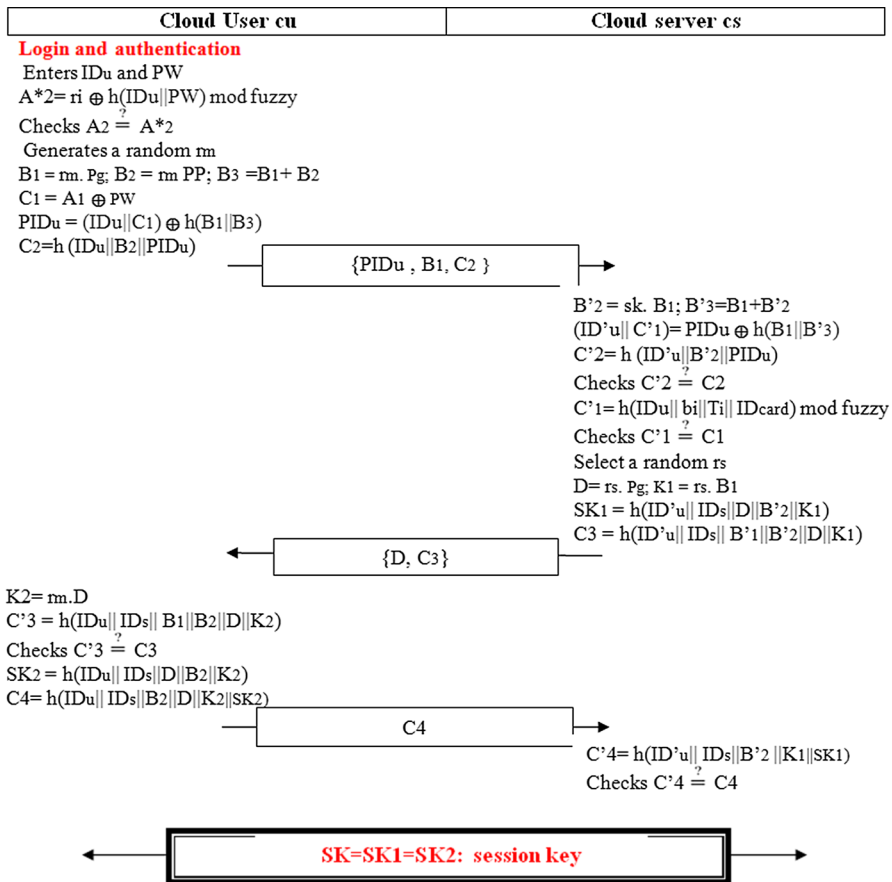


Fig. 2 Proposed login and authentication algorithm

- Step 2* The card should compute  $A_2^* = r_i \oplus h(ID_u \parallel PW) \text{ mod fuzzy}$ .
- Step 3* It compares  $A_2^*$  and  $A_2$  delivered by the cloud user. If equal, the user will be accepted. Else authentication process is canceled and an error message will be sent.
- Step 4* The card selects a random  $r_m \in Z_q^*$ . Three components should be computed  

$$: B_1 = r_m P_g, B_2 = r_m P_P, B_3 = B_1 + B_2, C_1 = A_1 \oplus PW, PID_u = (ID_u \parallel C_1) \oplus h(B_1 \parallel B_3) \quad \text{and}$$

$$C_2 = h(ID_u \parallel B_2 \parallel PID_u)$$
- Step 5* Cloud user constructs and sends a login request  $\{PID_u, B_1, C_2\}$ .

#### 4.5.2 Cloud server login response

The cloud server uses its secret key  $s_k$  to compute some values.

- Step 1* It determines  $B_2' = S_k B_1, B_3' = B_1 + B_2', (ID_u' \parallel C_1') = PID_u \oplus h(B_1 \parallel B_3')$   
and  $C_2' = h(ID_u' \parallel B_2' \parallel P_g ID_u)$
- Step 2* Cloud server checks if  $C_2' = C_2$ . If it is not, it aborts the current session.
- Step 3* Otherwise, it authenticates the user and finds the record  $\langle T_i, B_i, ID_{card} \rangle$ . The cloud server computes  $C_1' = h(ID_u \parallel b_i \parallel T_i \parallel ID_{card}) \text{ mod fuzzy}$ . Finally it checks  $C_1' = C_1$ . If it is not, it terminates the session
- Step 4* Else, it chooses a random  $r_s \in Z_q^*$  and sets  $D = r_s P_g, K_1 = r_s B_1, SK_1 = h(ID_u' \parallel ID_s \parallel B_2' \parallel D \parallel K_1)$  and  $C_3 = h(ID_u' \parallel ID_s \parallel B_1' \parallel B_2' \parallel D \parallel K_1)$
- Step 4* Cloud server sends  $\langle D, C_3 \rangle$  to cloud user CU.

#### 4.5.3 Mutual verification and session key negotiation

In this section, cloud user and cloud server will exchange and verify credentials.

- The user computes  $k_2 = r_m D, C_3' = h(ID_u \parallel ID_s \parallel B_1 \parallel B_2 \parallel D \parallel k_2)$
- He compares  $C_3'$  to  $C_3$ . If equal, he computes  $SK_2 = h(ID_u \parallel ID_s \parallel B_2 \parallel D \parallel K_2)$  and  $C_4 = h(ID_u \parallel ID_s \parallel B_2 \parallel D \parallel K_2 \parallel SK_2)$ . He sends only  $C_4$  to cloud server. Else, session is aborted.
- In turn, cloud server will compute  $C_4' = h(ID_u' \parallel ID_s \parallel B_2' \parallel K_1 \parallel SK_1)$ .
- If the received value and the computed one do not match, then session key is refused. Otherwise, it sets up  $SK_1 = SK_2 = h(ID_u \parallel ID_s \parallel B_2 \parallel D \parallel K_1)$  and the latter will be considered as the shared session key.

#### 4.6 Update phase

Update phase consists of modifying user password. The cloud user should insert the smart card and key  $(ID_u, PASS_u)$ .

- Step 1* The card computes  $A_2' = r_i \oplus h(ID_u \parallel (h(r_i \parallel PASS_u)))$
- Step 2* It checks  $A_2 = A_2'$ . If not, the request will be rejected else the user inputs a new password  $PASS^{new}$ .

*Step 3* The smart card replaces  $(A_1, A_2)$  by  $(A_1^{new}, A_2^{new})$ . To do so, it firstly computes  $PW^{new} = h(r_i \parallel PASS^{new})$ , then he sets  $A_1^{new} = A_1 \oplus PW' \oplus PW^{new}$  and finally, it computes  $A_2^{new} = r_i \oplus h(ID_u \parallel PW^{new}) \bmod \text{fuzzy}$ .

#### 4.7 Card revocation phase

For one reason or another, a cloud user can revoke its smart card. First, he must authenticate and send revocation request to the cloud server. This request should contain user ID,  $B_1$  and  $C_2$ . In turn, cloud server sets registration values to NULL. Strictly speaking, the registration time  $T_i$ , the random number  $b_i$  and the card number  $ID_{card}$ .

### 5 Security analyses

In this section, we will provide security validation formally and informally. In informal validation, we will use authentication key exchange model (AKE) and SCYTHYR tool. In informal validation, we will discuss some security requirements and adversary games.

#### 5.1 Formal validation

In this section, we develop formal validation based on AKE model and by using Scyther tool.

##### 5.1.1 Authentication Key Exchange (AKE) security model

In this section, we provide a formal security analysis of our solution based on a random oracle model [31]. We will adjust the security model proposed in [32] to adapt it to our scheme. All the participants are modelled as probabilistic polynomial time (PPT) Turing machines. The  $i^{th}$  instance of a participant P is denoted by  $\Pi$ . The adversary A can make the following oracle queries:

- Execute* This oracle query harvests the passive eavesdrop ability of the adversary. A will capture all the public transcripts of  $\Pi$ .
- Send* Contrary to the execute query, the latter can capture the active attack ability of the adversary. A injects a modified or forged message  $\mathbf{m}$  into an instance  $\Pi$ . The participant instance  $\Pi$  will generate a message  $\mathbf{m}+1$  after receiving the message  $\mathbf{m}$ . Consequently, A will capture it.
- Corrupt-pass* By this query, the adversary A can corrupt the cloud user's password. It cannot take control or compromise the victim credential.
- Corrupt-cred* In this case, A can extract and corrupt the credential delivered by the agent and control the user's terminal. It can have any idea about the password.

<i>Reveal</i>	This query is used to find out whether the instance $P_i$ has a session key or not. If the instance accepts the query, it outputs a session key and the adversary will corrupt it. Otherwise, it generates the symbol $\perp$ .
<i>Test</i>	This query tests the strength of the session key security in an instance $P_i$ . In turn, it sends a Boolean result. If “1”, the instance will return to the adversary a real session key ELSE the simulator generates a random session key. The adversary should guess if the received key is the current one or a random delivered from the simulator.

**Theorem 1** *Let  $S$  our proposed cloud computing authentication scheme and  $P$  the password space. Suppose that all passwords follow the Zipf's law as frequency distribution and  $A$  is a probabilistic polynomial time (PPT) adversary. We assume that  $A$  sends a huge number of send queries*

*$send_{max}$  with execution time  $t_{send}$ . The ability of an adversary to break AKE security of the above configuration is modeled as follows:  $ADV_{S,P}^{AKE}(A)$ .*

*Let's our scheme hash function behaves like a random oracle and the signature scheme is unforgeable against adaptive chosen message attacks. Under the difficult assumption of CDH problem, we have:*

$$ADV_{S,P}^{AKE}(A) \leq C \cdot send_{max}^l \cdot \epsilon(l) \quad (1)$$

**Proof** In order to proof theorem 1, we eject seven experiment  $EXP_i$  (with  $i=0,1,2,3,4,5,6$ ). In each experiment, the adversary  $A$  will launch a Test query to guess a real bit. The event of collecting a correct bit is denoted by  $EV_i$  and the corresponding probability is  $\Pr[EV_i]$ .  $\square$

- *Experiment 0* This experiment launches a real attack scenario under a random oracle model similar to [32] security evaluation model. Consequently, the adversary advantage is:

$$ADV_{S,P}^{AKE}(A) = Pr[EV_0] \quad (2)$$

- *Experiment 1* This experiment, we simulate the hash function  $h()$  and it elaborates a hash list  $List_h$ . Since hash function can be simulated perfectly in a probabilistic polynomial time, the probability of this experiment is indistinguishable from experiment 1. Thus, we have

$$|Pr[EV_1] - Pr[EV_0]| \leq \epsilon(l) \quad (3)$$

- *Experiment 2* In this experiment, we eliminate all sessions in which a collision event takes place. If random oracle queries can collude with current session during the simulation of hash function or transcript of the following outputs  $\{PID_u$

$\{B_1, C_2, C, C_3, C_4\}$ , the adversary A wins the game. To define the advantage of A, we call the birthday paradox. Consequently, we have:

$$|Pr[EV_2] - Pr[EV_1]| \leq \epsilon(l) \tag{4}$$

- *Experiment 3* In this experiment, we use a hash function  $h'()$  instead of  $h()$  in order to guess the session key in passive session. A passive session in which Diffie–Hellman keys are removed and collisions in hash function and transcripts are ruled out. In our case,  $K_1$  and  $K_2$  are removed from  $SK = h'(ID_u || ID_s || D || B_2 || D)$  and  $C_3 = h'(ID_u || ID_s || B_1 || B_2 || D)$ . The adversary A can distinguish the experiment 2 and 3 only if he can compute  $K_1$  and  $K_2$  and sends a query  $(ID_u || ID_s || D || B_2 || D || K_1)$  to oracle hash function. Seen that we have a CDH problem, this assumption is computationally hard.

**Proof** Given a CDH instance (A,B), we can use the self-reducibility problem to simulate passive session. Let's  $a_0, b_0, a_1, b_1 \in Z_p^*$ , for each passive session we set  $X = a_0A + b_0P_g$  and  $Y = a_1B + b_1P_g$  where ( $P_g$  is the generator of an additive cyclic group). If the adversary A can deliver  $\frac{K_1 - (a_0b_1A) - (a_1b_0B) - (b_0b_1P_g)}{a_0a_1}$  as an answer to (A,B) and distinguish the experiment 3 and 2 with non-negligible advantage we have :

$$|Pr[EV_3] - Pr[EV_2]| \leq \epsilon(l) \tag{5}$$

□

- *Experiment 4* In this case, an adversary have to guess the secret key  $k_2$  in order to impersonate the cloud user. Consequently, it will construct  $C'_4 = h(ID_u || ID_s || K_2 || SK_2)$  (we speak about an active session). If  $C'_4$  and  $C_4$  are equals, the cloud server will return a record  $(\{PID_u, B_1, C_2\}, \{D, C_3\}, \{c_4\})$ . Due to the hardness of CDH problem, we have:

$$|Pr[EV_4] - Pr[EV_3]| \leq \epsilon(l) \tag{6}$$

- *Experiment 5* In this experience, an adversary will handle the active session and guess  $k_1$  without asking the hash query. He will send to the cloud user the values of  $(D, C_3)$ . To do so, we modify the sent query and we search in the hash list any record  $(\star || ID_u || \star || k_1)$ . If the result is  $\perp$  the game finishes else the session key and  $C_3$  are computed as follows:  $SK = h(ID_u || ID_s || B_2 || D || k_1)$ ,  $C_3 = h(ID_u || ID_s || B_1 || B_2 || D || k_1)$ . If the adversary A can guess the secret  $k_1$  without using the hash function  $h()$ , he wins with the probability of:

$$|Pr[EV_5] - Pr[EV_4]| \leq \epsilon(l) \tag{7}$$

- *Experiment 6* In the last experiment, we modify another time the send query. When Send (CS,  $\{PID_u, B_1, C_2\}$ ) query is ejected, the cloud server should compute  $ID_u, C_1, C'_2, B_2$  and  $B_3$ . Then, it verifies that  $C'_2$  and  $C_2$  are equals. If the sub-

mitted query is correct, the adversary A wins. Based on previous experiment, we announce that the success probability of forging authenticator  $C_2$  is negligible.

$$|Pr[EV_6] - Pr[EV_5]| \leq \epsilon(l) \tag{8}$$

In the sixth experiment, the only possibility for the adversary A to win the game is to corrupt the smart card and guess the password. Thanks to Zipf’s law, the adversary has no advantage to obtain the user password. As consequence,

$$Pr[EV_6] \leq C.send_{max}^c \tag{9}$$

Finally, the theorem 1 is proved by equations (2)-(9).

**Theorem 2** *Let’s S the proposed scheme and A a probabilistic polynomial time (PPT) adversary. A wins to break the anonymity of S with an advantage of:*

$$ADV_{\mathbf{P}}^{ANONYM}(A) \leq \epsilon(l) \tag{10}$$

**Proof** Suppose that A can break the anonymity of the proposed scheme with a non-negligible advantage under the assumption of CDH problem.

Based on registration algorithm, we select  $r_m, r_{test} \in Z_q^*$  and we input  $(P_g, r_m P_g, s_k P_g, r_m s_k P_g)$  and  $(P_g, r_m P_g, s_k P_g, r_{test})$  with  $s_k$  is the private key of the cloud server. Suppose that cu is a legitimate user with a valid smart card and password and exchanges some authentication information with the cloud server (as illustrated in the authentication and registration algorithm in section 4.2).

- Step 1* In the first session  $Skey_{cu}^i$ , the cloud user computes and sends  $B_1(cu) = r_m P_g, B_2(cu) = r_m s_k P_g$ .
- Step 2* In the second session  $Skey_{cu}^j$ , the cloud user computes and sends  $B_1(cu) = r_m P_g, B_2(cu) = r_{test}$ . To make these session keys uniform and have the same structure, a user can choose  $r_s \in Z_q^*$  and select two random bit strings for  $C_3$  and  $C_4$ .
- Step 3* In third time, we select a random  $r'_m$ , compute  $B_1(cu) = r'_m P_g, B_2(cu) = r'_m s_k P_g$  and we have a session denoted  $Skey_{cu}^k$ .
- Step 4* The adversary A will check the anonymity of  $\{Skey_{cu}^k, Skey_{cu}^j\}$  and  $\{Skey_{cu}^i, Skey_{cu}^k\}$ . He launches a test anonymity query and returns two bits x and y. These bits can have different combinations:

- $x=0$  and  $y=0 \Rightarrow$  Any Diffie-Hellman tuple is found.
- $x=0$  and  $y=1 \Rightarrow B_1$  and  $B_2$  of the first session  $Skey_{cu}^k = i_{cu}$  presents a Diffie–Hellman tuple.
- $x=1$  and  $y=0 \Rightarrow B_1$  and  $B_2$  of the second session  $Skey_{cu}^j$  presents a Diffie–Hellman tuple.
- $x=1$  and  $y=1 \Rightarrow$  both are Diffie-Hellman tuples.

In this scenario  $S_k P_g$  is fixed and  $r_m P_g$  changes in every launch of this protocol. Based on the self-reducibility of CDH problem, we have

$$ADV_{CDH,S} = |Pr[cu(P_g, r_m P_g, s_k P_g, r_m s_k P_g)] - Pr[cu(P_g, r_m P_g, s_k P_g, r_{test})] = 1| \tag{11}$$

As consequence, we obtain:  $ADV_{CDH,S} \geq |Pr[ANONYM(Skey_{cu}^i, Skey_{cu}^j) = 1] - Pr[ANONYM(Skey_{cu}^i, Skey_{cu}^k) = 1]|$  As a conclusion, the adversary A has to solve the CDH problem to break untraceability. But resolving this problem is computationally hard. Therefore, the theorem 2 is proved. □

### 5.1.2 Formal security verification and validation with Scyther tool

In this section, we analyze the security of our scheme using a formal analysis security protocols based on “Scyther tool” [33]. Scyther offers a simple graphic user interface incorporating the scyther command line and python interface. It requires protocol description and parameters and outputs reports. It also visualizes for each attacks a detailed graph. The description and detail of a scenario is written in SPDL language.

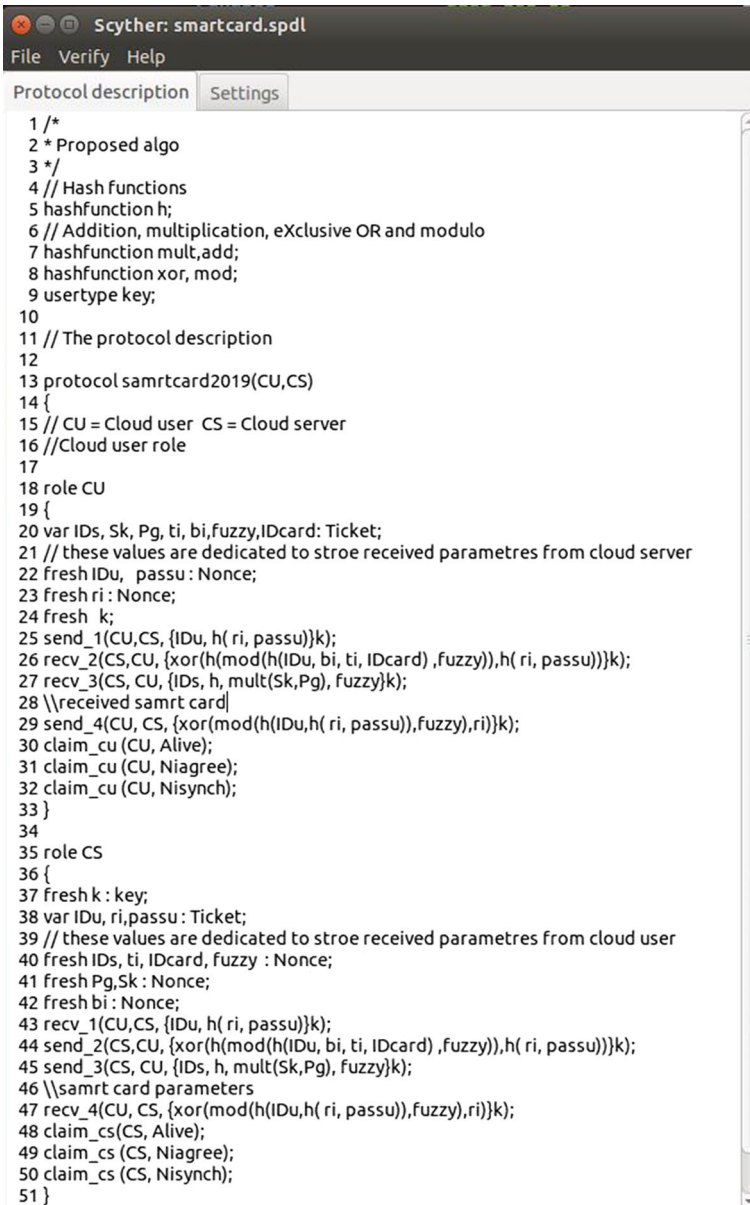
In scyther tool, three protocol verification ways are possible:

- *Automatic claims* scyther tool possesses a default claim patterns. If any claim events is defined, scyther will automatically generates and verifies claims.
- *Verification claim* if a user defines a set of security properties. Scyther will just checks the falsifiers claims.
- *Characterization* each protocol role can be characterized.

In Scyther tool, the security specifications are expressed in the form of “claims”. We use in our simulation the list below:

- *Secret* ensures that an entity X has kept secret a key  $x \mapsto \text{claim}(X, \text{Secret}, x)$ . If, in addition, we want the key  $x$  to be a session key, we should check with the SKR (Session Key Reveal) property  $\mapsto \text{claim}(X, \text{SKR}, x)$
- *Alive* allows to check the authenticity of an entity  $X \mapsto \text{claim}(X, \text{Alive})$ .
- *Nisynch* verifies that all messages sent by the sender X are received by the receiver Y in the appropriate order  $\mapsto \text{claim}(X, \text{Nisynch})$
- *Niagree* ensures that the execution of a protocol between two entities X and Y is completed and that the contents of the messages are the same at X and at Y  $\mapsto \text{claim}(X, \text{Niagree})$
- *Running* specifies a data agreement between two entities X and Y on a parameter or key  $x \mapsto \text{claim}(X, \text{Running}, Y, x)$

In Fig. 3, we model our protocol in Security Protocol Description Language (SPDL). We define two entities namely cloud user CU and cloud server CS. Each entity is represented by a separated role. Figure 4 shows the set of claims that we have defined so that the Fig. 5 illustrates the set of automatic launched claims. It is clear in both cases that our proposed solution guarantees all security properties. First, all



```

Scyther: smartcard.spdl
File Verify Help
Protocol description Settings
1 /*
2 * Proposed algo
3 */
4 // Hash functions
5 hashfunction h;
6 // Addition, multiplication, exclusive OR and modulo
7 hashfunction mult, add;
8 hashfunction xor, mod;
9 usertype key;
10
11 // The protocol description
12
13 protocol samrtcard2019(CU, CS)
14 {
15 // CU = Cloud user CS = Cloud server
16 // Cloud user role
17
18 role CU
19 {
20 var IDs, Sk, Pg, ti, bi, fuzzy, IDcard: Ticket;
21 // these values are dedicated to stroe received parametres from cloud server
22 fresh IDu, passu: Nonce;
23 fresh ri: Nonce;
24 fresh k;
25 send_1(CU, CS, {IDu, h(ri, passu)}k);
26 rcv_2(CS, CU, {xor(h(mod(h(IDu, bi, ti, IDcard), fuzzy)), h(ri, passu))}k);
27 rcv_3(CS, CU, {IDs, h, mult(Sk, Pg), fuzzy}k);
28 \\received samrt card
29 send_4(CU, CS, {xor(mod(h(IDu, h(ri, passu)), fuzzy), ri)}k);
30 claim_cu(CU, Alive);
31 claim_cu(CU, Niagree);
32 claim_cu(CU, Nisynch);
33 }
34
35 role CS
36 {
37 fresh k: key;
38 var IDu, ri, passu: Ticket;
39 // these values are dedicated to stroe received parametres from cloud user
40 fresh IDs, ti, IDcard, fuzzy: Nonce;
41 fresh Pg, Sk: Nonce;
42 fresh bi: Nonce;
43 rcv_1(CU, CS, {IDu, h(ri, passu)}k);
44 send_2(CS, CU, {xor(h(mod(h(IDu, bi, ti, IDcard), fuzzy)), h(ri, passu))}k);
45 send_3(CS, CU, {IDs, h, mult(Sk, Pg), fuzzy}k);
46 \\samrt card parameters
47 rcv_4(CU, CS, {xor(mod(h(IDu, h(ri, passu)), fuzzy), ri)}k);
48 claim_cs(CS, Alive);
49 claim_cs(CS, Niagree);
50 claim_cs(CS, Nisynch);
51 }

```

Fig. 3 SPDL code of registration algorithm



Scyther results : verify					
Claim				Status	Comments
samrtcard2019	CU	samrtcard2019,cu	Alive	ok	No attacks within bound
		samrtcard2019,CU1	Niagree	ok	No attacks within bound
		samrtcard2019,CU2	Nisynch	ok	No attacks within bound
	CS	samrtcard2019,cs	Alive	ok	No attacks within bound
		samrtcard2019,CS1	Niagree	ok	No attacks within bound
		samrtcard2019,CS2	Nisynch	ok	No attacks within bound
Done.					

Fig. 4 User claims

Scyther results : autoverify					
Claim				Status	Comments
samrtcard2019	CU	samrtcard2019,CU1	Secret k	ok	No attacks within bounds.
		samrtcard2019,CU2	Secret ri	ok	No attacks within bounds.
		samrtcard2019,CU3	Secret passu	ok	No attacks within bounds.
		samrtcard2019,CU4	Secret IDu	ok	No attacks within bounds.
		samrtcard2019,CU5	Secret IDcard	ok	No attacks within bounds.
		samrtcard2019,CU6	Secret fuzzy	ok	No attacks within bounds.
		samrtcard2019,CU7	Secret bi	ok	No attacks within bounds.
		samrtcard2019,CU8	Secret ti	ok	No attacks within bounds.
		samrtcard2019,CU9	Secret Pg	ok	No attacks within bounds.
		samrtcard2019,CU10	Secret Sk	ok	No attacks within bounds.
		samrtcard2019,CU11	Secret IDs	ok	No attacks within bounds.
		samrtcard2019,CU12	Alive	ok	No attacks within bounds.
		samrtcard2019,CU13	Weakagree	ok	No attacks within bounds.
		samrtcard2019,CU14	Niagree	ok	No attacks within bounds.
		samrtcard2019,CU15	Nisynch	ok	No attacks within bounds.

Fig. 5 Automatic claims

exchanged values and keys are proved secure. Second, it guarantees the aliveness of the proposed nodes, Non-injective synchronization (Nisynch), authentication and traceability, and Non-injective agreement (Niagree).

### 5.2 Informal validation

In order to achieve a robust and efficient authentication solution, many security requirements should be verified. Therefore, we compare in Table 2 various security properties against our solution and related authentication mechanisms. We will also discuss in-depth some security requirements as follows:

- *Mutual authentication* we design by mutual authentication the fact that cloud server and cloud user checks and verifies the identity of each other. This propriety is assumed based on  $C_2$  and  $C_3$  values. The cloud server checks and verifies the validity of  $C_2$  component and authenticate the cloud user if it passes the test. In the opposite way, the cloud user will verify the legitimacy of a server by checking  $C_3$ .
- *Privacy* our scheme ensures both anonymity and non traceability. These two properties are the basis of privacy definition [34]. In our solution, anonymity and privacy are verified by the fact that the attacker cannot reveal the identity of the sender and cannot distinguish whether the set of stolen messages are delivered from the same user. To compute the  $PID_u$ , the card should at each session compute the hash function of  $B_1$  and  $B_3$ . Notice that  $B_3$  value changes in each session due to the change of the random  $r_m$  ( $PID_u = (ID_u \parallel C_1) \oplus (B_1 \parallel B_3)$ ). Thus, an attacker cannot retrieve the identity of the cloud user because he/she has no knowledge of  $r_m$  and  $s_k$ .
- *Replay attacks* to withstand with this issue, both cloud user and cloud server compute different values generated by random number mechanism, respectively  $r_m$  and  $r_s$ .
- *Forward secrecy* The session key  $SK_1 = SK_2 = h(ID_u \parallel ID_s \parallel B_2 \parallel D \parallel K_2)$  is generated based on partial components ( $B_2 = r_mPP$ ,  $D = r_sP_g$ ,  $K_2 = r_mD$ )

**Table 2** Security comparison

Schemes	Our	[16]	[14]	[21]	[15]	[23]	[17]	[18]	[19]
User anonymity	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes
Key-agreement phase	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Prevent insider attack	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes
Prevent replay attack	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Perfect forward secrecy	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Perfect mutual authentication	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Prevent user impersonation attack	Yes	Yes	No	Yes	No	No	Yes	No	Yes
Prevent password-guessing attack	Yes	Yes	No	No	Yes	Yes	No	No	Yes
Prevent smart card stolen attack	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes

$=r_m r_s P_g$ ) delivered simultaneously by cloud user and server. Although an attacker intercepts  $P_g$ , PP and D from the public channel, he cannot compute  $B_2$  or  $B'_2$  components because they are one of the randoms  $r_m/r_s$  or the session key  $s_k$ . This assumption is hard based on elliptic curve discrete logarithm and computational Diffie–Hellman problem.

- *Privileged insider attack* In the registration phase, a user send  $\{ID_u, h(r_i || PASS_u)\}$  to cloud remote sever. Even an attacker can intercept this message, he cannot retrieve the password because it is blinded by the value of  $r_i$  such the password is computed as follows:  $(PW = h(r_i || PASS_u))$ .
- *Verifier-stolen attack* In our solution, we store  $(ID_u, b_i, T_i$  and  $ID_{card})$  in cloud server. If an attacker compromises this table, any attack can be launched. This is due to the fact that these records are non security-related.
- *Known-key security* It means that an attacker cannot calculate the current session key even though he has some previous session keys. In our scheme, the session key  $Sk = h(ID'_u || ID_s || \bar{B}_2 || B || K_1 = h(ID'_u || ID_s || \bar{B}_2 || D || K_2)$ . Seeing that we use one-way hash function, an attacker cannot takeout  $(B_2, K_2)$  or  $(B'_2, K_1)$  from the session key. Consequently, disclosure of some old session key does not result in the revelation of current session key.
- *Offline dictionary attack* When the smart card of the cloud user is lost or stolen, an attacker can have some secret information by launching side-channel attack [35]. He can reveals  $A_1, A_2, ID_s, PP, P_g, h()$  and the master key  $s_k$ . Here, an adversary can launches dictionary attack on two ways:

**\* The adversary will use  $A_2$  to launch the attack**

1. He selects from the dictionary space of the user identity and password a pair  $(ID_u^*, PW_i^*)$  to start the game.
2. The adversary executes the steps in authentication phase. Namely he computes  $PW^* = h(r_i || PASS_u^*)$  and  $A_2^* = r_i \oplus (h(ID_u || h(r_i) || PW^*) \text{ mod } m)$ .
3. He checks the correctness of these credentials by verifying if  $A_2$  and  $A_2^*$  are equal. If it holds, the divined pair (ID,password) is considered correct. Otherwise, the adversary will repeat steps (1)-(3) until he falls in a correct pair or leaves.

However, the adversary fails to find this assumption because the identity and the password are strongly dependent on  $r_i$  which is large enough to prevent the adversary from guessing the value. Even if he can find the value of  $r_i$  satisfying these equations, fuzzy verifier mechanism brings up an important number of pairs  $(ID_u^*, PASS_u^*)$  up to  $2^{64}$ . It is large enough to make this attack unattainable.

**\* The adversary will use  $C_2$  to launch the attack**

Even if the adversary success to guess  $C_2$  and obtains  $ID_u$  from  $h(ID_u || B_2 || PID_u)$ , he cannot calculate  $B_2 = r_m PP$  because he has no idea about the Nonce  $r_m$ . Therefore, this attack cannot be successful.

- *Fraud* In order to prevent fraud, we implement in the proposed solution a mutual authentication process. The cloud server and cloud user should authenticate each others.

- *User impersonation attack* If the adversary want to impersonate the cloud server, he should guess either the pair  $(ID_u, Pass_u)$  or the login request  $\{PID_u, B_1, C_2\}$ . Furthermore, if he/she selects a random  $r_{attack}$  and computes  $B1_{attack} = r_{attack} \cdot Pg$ , he/she forges  $PID_{attack}$  and  $C2_{attack}$  and constructs the login request  $\{PID_{attack}, B1_{attack}, C2_{attack}\}$ . However, the cloud sever cannot extract the appropriate user ID  $ID_u$  from the table. We add also that when he tries to compute  $C'2_{attack}$ , the result will be different from the received value. As consequence, our solution can withstand this attack.

## 6 Simulation experiments

We develop our proposal with java programming language. The experiments are performed using a machine with an Intel Core i7 processor 8th generation and 16 GB total memory. We call Java Pairing-Based Cryptography Library (JPBC) and Java card development kit in particular javax.smartcardio API. We have created three data centers. Each unit is made up of two powerful servers of the hp ProLiant ML110 g5 type. Each server contains 50 homogeneous AWS (EC2) instances.

In order to evaluate the efficiency of the proposed solution, we provide in the first time a comparison with related schemes in terms of computational and communication cost. Some competitive works used in this section are [14–19, 21, 23]. All the simulation results are retrieved among a mean of 100 trials.

Our basic goal is to improve security of the authentication scheme. In some cases, it requires a little bit increase in simulation metrics. We mention that all simulation experiments are only in login and authentication phase. We will not take into consideration registration, update and card revocation phases because they are respectively onetime or sometimes process. The following notations are used:

- $T_m$  : the time cost of a point multiplication
- $T_a$  : the time cost of a point addition
- $T_h$  : the time cost of a hash operation
- $l_{id}$  : the length of an identity
- $l_{point}$  the length of a point
- $l_{hash}$  : the length of a one-way hash value
- $l_t$  : the length of a timestamp

Based on the study [36], the time of performing a concatenate operation and an Exclusive-OR operation (XOR) can be negligible because they are much less than a hash function.

**Table 3** Computation cost in cloud user and cloud server sides

Schemes	Cloud user side	Cloud server side
[16]	$4T_m + 1T_a + 8T_h$	$5T_m + 1T_a + 10T_h$
[14]	$7T_m + 3T_a + 10T_h$	$5T_m + 2T_a + 6T_h$
[21]	$4T_m + 4T_h$	$5T_m + 5T_h$
[15]	$5T_m + T_a + 10T_h$	$4T_m + 8T_h$
[23]	$2T_m + 7T_h$	$2T_m + 8T_h$
[17]	$4T_m + 10T_h + 1T_a$	$5T_m + 9T_h$
[19]	$6T_m + 15T_h$	$6T_m + 8T_h$
[18]	$4T_m + 10T_h$	$3T_m + 9T_h$
Our solution	$3T_m + T_a + 6T_h$	$3T_m + T_a + 6T_h$

**Table 4** Different operation's cost

Execution time	$T_m$	$T_a$	$T_h$
Cloud user	0.11 s	0.09 s	0.001 s
Cloud server	1.16 ms	0.087 ms	0.0009 ms

### 6.1 Computation cost

Total computation cost is the sum of the computation cost in cloud user and server.

Most of the related work do not differentiate between the computation cost in the user side and in the server side. In our evaluation, we study computation cost in each entity separately in order to be able to judge the robustness and efficiency of our solution. We assume that in the context of cloud computing, the operations of multiplication, hashing as well as addition have different costs in server and user side. This is due to the different configuration of the hardware devices. The main objective is to provide a secure, mutual but above all lightweight authentication and key agreement solution to be adapted to actual requirement in which a cloud user can accomplish different tasks with mobile devices.

Table 3 shows the different operations performed by each entity separately . Computation cost is the sum of different operations of addition, multiplication and basically hash functions. Table 4 reveals the execution time of these operations in our simulation environment in both user and server side. We deduce that the powerful servers execute the operations in terms of ms while the client finishes different tasks in approximately one second. Table 5 computes execution time in different related works both in user and in server sides.

### 6.2 Communication overhead

In our solution, we set the length of metrics used in simulation environment as illustrated in Table 6. We compute for the proposed solution and related works the number of exchanged messages, the length of communication messages and the communication cost. Table 7 shows that the discussed schemes exchange mainly three

**Table 5** Execution time in cloud user and cloud server sides

Schemes	Cloud user side (s)	Cloud server side (ms)
[16]	0.61	5.896
[14]	1.05	5.979
[21]	0.444	5.804
[15]	0.647	4.647
[23]	0.227	2.327
[17]	0.54	5.8081
[19]	0.675	6.967
[18]	0.45	3.448
Our solution	0.426	3.57

**Table 6** Simulation parameters

Parameters	$l_{id}$	$l_{point}$	$l_{hash}$	$l_t$
Length (bits)	32	1024	160	32

**Table 7** Communication cost

Schemes	communication cost	Communication length (bits)	Number of messages
[16]	$5 l_{hash} + 2 l_{point}$	2848	3
[14]	$4 l_{hash} + 3 l_{point}$	3712	3
[21]	$2 l_t + l_{id} + 3 l_{hash} + 2 l_{point}$	2624	3
[15]	$7 l_{hash} + 2 l_{point}$	3168	3
[23]	$1 l_{id} + 5 l_{hash} + 4 l_{point}$	4928	4
[17]	$3 l_{hash} + 2 l_{point} + 3 l_t$	2624	3
[19]	$4 l_{hash} + 2 l_{point} + 2 l_t + 2 l_{id}$	2848	2
[18]	$4 l_{hash} + 2 l_{point} + 1 l_{id}$	2720	3
Our solution	$4 l_{hash} + 2 l_{point}$	2688	3

messages in the authentication phase with the exception [19, 23]. The total length of these messages is expressed in bits. We can deduce with the same number of exchanged messages we can obtain different communication length. In view of the highest length of a point, we have worked in our solution to minimize the number of this operation.

### 6.3 Discussion

In this section, we will discuss the security, computation cost and communication overhead results of the proposed solution.

First of all, we can observe that in term of computation cost, our scheme requires a half of second in user side and 3.57 ms in server side based on Table 5 and Figs. 6

Fig. 6 User side

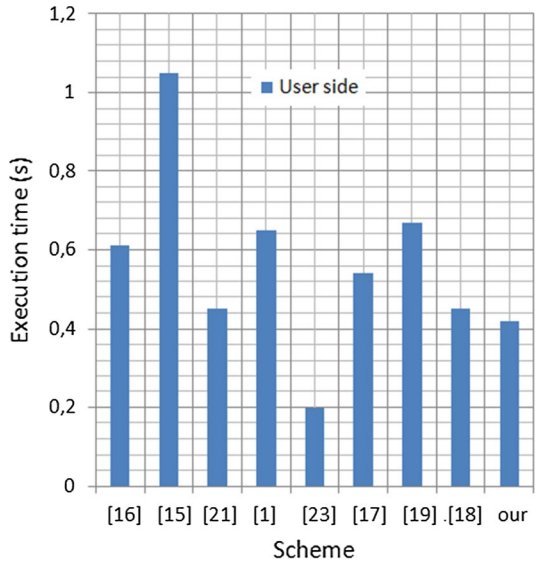
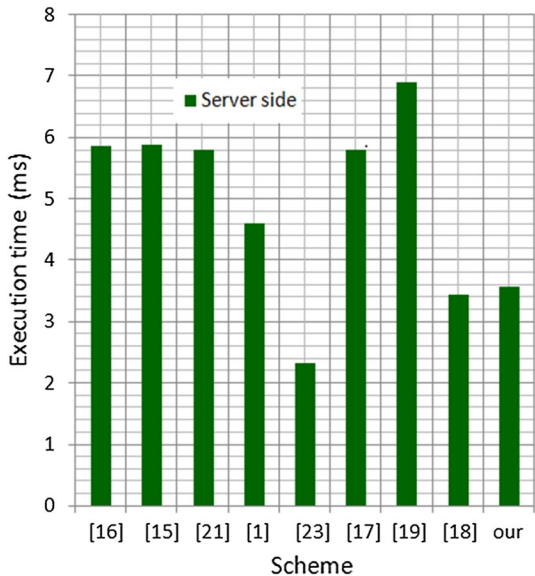


Fig. 7 Server side



and 7. Our solution is just slightly higher than what is obtained in [23] in both user and server sides, has extra time compared to [18] only on the server side and it is much less than solutions proposed in [14–17, 19, 21] in both user and server sides. From the discussion above, we can confirm that our solution is efficient and presents the best computation cost comparing to most of the related works.

Thereafter, we note from Table 7 that our solution consumes 336 bytes in communication process. We can see that it outperforms schemes [14–19, 23]. Works

discussed in [21] and [17] present a communication overhead of eight-bytes less compared to the proposed solution.

We can also deduce from the aforementioned table that the number of exchanged messages in the login and authentication is three and equal in all schemes except [23] and [19]. From this study, we can note that the more the number of messages, the more the communication overhead becomes greater. We add also that by keeping the same number of exchanged messages, communication length differs from one solution to another and this is explained by the fact that it depends on the type of operations. In brief, we success to keep the same number of messages and to reduce the communication overhead in our work.

Figure 8 shows the communication cost of the related works and proposed solution incrementally. It shows that compared to similar proposals, our solution is located in third position after [21] and [17] that do not meet several security properties. From this side, the proposed solution can be considered efficient and it offers acceptable results.

Finally, we are going back to security evaluation presented in Table 5. We will study the relation between security and performance evaluation. We start by solution proposed in [23] which outperforms the proposed scheme in terms of computation cost. We mention that this solution does not ensure mutual authentication. We add also that it does not resist to insider and user impersonation attacks. Move to scheme [21] that have better communication overhead. This solution does not establish user anonymity and untraceability. It suffers also from password-guessing attack and smart card loss/stolen attack. Now, we can understand why our solution presents a little bit increase in performance metrics. For the rest of related works, our scheme outshines in terms of security and performance.

In summary, based on this discussion, we can draw a conclusion that our solution is in the first hand efficient in terms of communication overhead and computation cost. In the second hand, it withstands various attacks and more secure

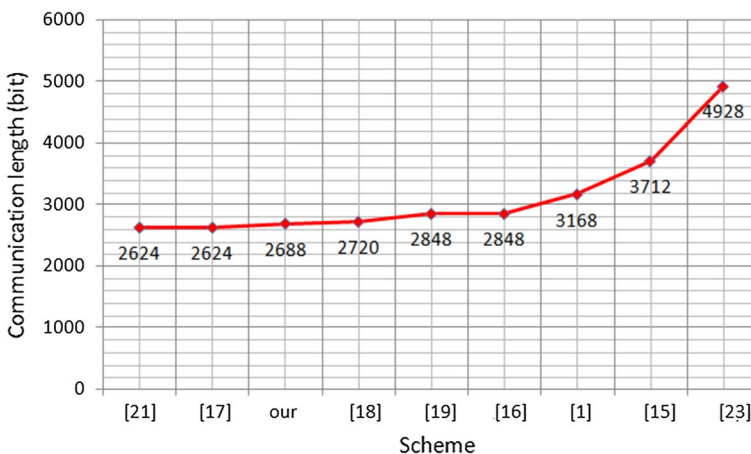


Fig. 8 Communication cost



compared to related works. Finally, it present a lightweight, anonymous, mutual and secure two-factor authentication and key agreement scheme which can be deployed in the context of cloud computing environment.

## 7 Conclusion

This paper proposes an efficient and secure authentication and key agreement solution for cloud computing. This contribution presents a marriage of approaches, in particular, fuzzy verifier, Elliptic Curve Cryptography, two-factor authentication, etc. The formal validation with scyther tool and informal validation proves that our solution can withstand to various attacks, such as impersonation attack, privileged insider attack, replay attack and others. Based on random oracle model, the proposed scheme is proved provably secure under CDH and ECDL problem. The performance evaluation shows that our solution shows that our solution is efficient and robust compared to related works. As a perspective, we will be using temporary identities and trusted parties to get rid of the assumption that the cloud user must be honest.

## References

1. Boyko V, MacKenzie P, Patel S (2000) Provably secure password-authenticated key exchange using diffie-hellman. *International Conference on the Theory and Applications of Cryptographic Techniques* 1807:156–171
2. Lin C-L, Hwang T (2003) A password authentication scheme with secure password updating. *Comp Secur* 22(1):68–72
3. Peyravian M, Jeffries C (2006) Secure remote user access over insecure networks. *Comp Commun* 29(5):660–667
4. Merdassi I, Bouchaala M, Ghazel C, Leila S (October 2019) Private security for the cloud mobile via a strong authentication method. *Coop Design Vis Eng*, pages 190–200
5. Cherdmuangpak N, Anusas-amonkul T, Limthan B (July 2017) Two factor image-based password authentication for junior high school students. *International Joint Conference on Computer Science and Software Engineering (JCSSE)*
6. Das R, Manna S, Dutta S (2018) Secure user authentication system using image-based otp and randomize numeric otp based on user unique biometric image and digit repositioning scheme. *Commun Devices Comput*. pages 83–93
7. Trupil L, Nishant D (2017) An analytical study of biometric based remote user authentication schemes using smart cards. *Comput Electr Eng* 59:305–321
8. Das ML (2009) Two-factor user authentication in wireless sensor networks. *IEEE Trans Wirel Commun*, pages 1086 – 1090
9. Lee CC, Li CT, Der Chen S (2011) Two attacks on a two-factor user authentication in wireless sensor networks. *Parallel Process Lett*, pages 21–26
10. He D, Gao Y, CHAN S (2010) An enhanced two-factor user authentication scheme in wireless sensor networks. *Ad Hoc Sens Wirel Netw*, page 361–371
11. Preeti C, Hari O (2018) An efficient two-factor remote user authentication and session key agreement scheme using rabin cryptosystem. *Arab J Sci Eng* 43(2):661–673
12. Morteza N, Reza J, Hamed A (2017) A lightweight authentication and key agreement protocol preserving user anonymity. *Multimed Tools Appl* 76(11):13401–13423
13. Trupil L, Mukesh S, Kumar MS (2018) Advanced formal authentication protocol using smart cards for network applicants. *Comput Electr Eng* 66:50–63

14. Qu J, Tan XL (2014) Two-factor user authentication with key agreement scheme based on elliptic curve cryptosystem. *J Electr Comput Eng*
15. Huang B, Khan MK, Libing W, Muhaya Fahad T, He BD (2015) An efficient remote user authentication with key agreement scheme using elliptic curve cryptography. *Wirel Personal Commun* 85:225–240
16. Chaudhry SA, Naqvi H, Mahmood K (2017) An improved remote user authentication scheme using elliptic curve cryptography. *Wirel Pers Commun*, pp 1–19
17. Maitra T, Obaidat Mohammad S, Hafizul Islam SK (2016) Security analysis and design of an efficient ecc-based two-factor password authentication scheme. *Secur Commun Netw* 9:4166–4181
18. Chenyu W, Wang D, Guoai X, Guo Y (2017) A lightweight password-based authentication protocol using smart card. *Int J Commun Syst* 30(16):336
19. Srinivas J, Vinod K, Adesh K (2019) Eseap: Ecc based secure and efficient mutual authentication protocol using smart card. *J Inf Secur Appl* 51:1–19
20. Chou CH, Tsai KY, Chung-Fu L (2013) Two id-based authenticated schemes with key agreement for mobile environments. *J Supercomput* 66(2):973–988
21. Sabzinejad Farash Mohammad, Ahmadian Attari Mahmoud (2014) A secure and efficient identity-based authenticated key exchange protocol for mobile client-server networks. *J Supercomput* 69(1):395–411
22. Yanrong L, Li L, Peng H, Yang Y (2017) An anonymous two-factor authenticated key agreement scheme for session initiation protocol using elliptic curve cryptography. *Multimed Tools Appl* 76:1801–1815
23. Memon Imran, Hussain Ibrar, Akhtar Rizwan, Gencai Chen (2015) Enhanced privacy and authentication: an efficient and secure anonymous communication for location based service using asymmetric cryptography scheme. *Wirel Pers Commun* 84(2):1487–1508
24. Alavalapati Goutham Reddy, Ashok Kumar Das, Yoon Eun-Jun (2016) A secure anonymous authentication protocol for mobile services on elliptic curve cryptographys. *IEEE Access* 4:4394–4407
25. Xie Qi, Wong Duncan S, Wang Guilin (2017) Provably secure dynamic id-based anonymous two factor authenticated key exchange protocol with extended security model. *IEEE Trans Inf Forensics Secur* 12:1382–1392
26. Shehzad AC, Husnain N, Taeshik S (2015) Cryptanalysis and improvement of an improved two factor authentication protocol for telecare medical information systems. *J Med Syst*. pp 1801–1815
27. Sheetal K, Sood Sandeep K (2015) Secure authentication scheme for iot and cloud servers. *Pervasive Mobile Comput* 24:210–223
28. Sharma G, Kalra S (2015) A lightweight multi-factor secure smart card based remote user authentication scheme for cloud-iot applications. *J Inf Secur Appl* 42:95–106
29. Messerges TS, Dabbish EA, Sloan RH (2002) Examining smart-card security under the threat of power analysis attacks. *IEEE Trans Comput* 51:541–552
30. Amin Ruhul, Hafizul Islamb SK, Biswas GP (2016) Design of anonymity preserving three-factor authenticated key exchange protocol for wireless sensor network. *Comput Netw* 101:42–622
31. Bellare M, Rogaway P (1993) Random oracles are practical: a paradigm for designing efficient protocols. *ACM Conference on Computer and Communications Security*, pp 62–73
32. Wei F, Vijayakumar P, Qi J, Zhang R (2018) A mobile intelligent terminal based anonymous authenticated key exchange protocol for roaming service in global mobility networks. *IEEE Trans Sustain Comput* 5(2):2377–3782
33. s Examining smart-card sMauwL. Operational semantics and verification of security protocols. *Information Security and Cryptography series*, Springer (2012)
34. Debiao H, Neeraj K, Khurram KM (2018) Efficient privacy-aware authentication scheme for mobile cloud computing services. *IEEE Syst J* 12(2):1621–1631
35. Nilesh C, Anand Vijay S, Samrat M (2019) Towards identifying and preventing behavioral side channel attack on recording attack resilient unaided authentication services. *Comput Secur* 84:193–205
36. Koblitz N, Menezes A, Vanstone S (2000) The state of elliptic curve cryptography. *Designs Codes Cryptogr* 19(2–3):173–193